Burrows, 5-6.

> "Unfortunately, this procedure reads and decompresses a full compression block even if the caller wanted only some smaller unit, such as a file system block or a physical sector. We alleviate this problem by caching the entire decompressed block in memory, rather than just caching the requested sectors. The data could be placed in the file system buffer cache, but for simplicity in our prototype, we cached the last decompressed block within the read routine. Sprite LFS reads files sequentially in 4 KByte units, so this simple caching strategy typically achieves three hits for each 16 KByte compression block when reading large files.
>
> When the file system is reading non-sequentially, the additional time to read a full compression block cannot be hidden by caching. Fortunately, this time is small compared to the rotational latency. The time needed to decompress the full block in software is several milliseconds; it would be much smaller if decompression were implemented in hardware."

Burrows, 6.

2. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system.	Burrows, as evidenced by the example citations below, discloses "said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claims 1.1, 1.4, and 1.5 above.	

3. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system.	Burrows, as evidenced by the example citations below, discloses "said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Burrows discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> "A second issue is that applications often commit small amounts of data to disk, resulting in poor compression."

Burrows, 9.

> "The scheme benefits from concentrating on executable files, which are read by few things besides the operating system itself; no attempt is made to make the compression transparent to other applications."

Burrows, 18.

4. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an application program and an operating system of said computer system.	Burrows, as evidenced by the example citations below, discloses "said decompressed at least a portion of said boot data comprises program code associated with an application program and an operating system of said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above. *See also* "A second issue is that applications often commit small amounts of data to disk, resulting in poor compression." Burrows, 9. "The scheme benefits from concentrating on executable files, which are read by few things besides the operating system itself; no attempt is made to make the compression transparent to other applications." Burrows, 18.	

5. The method of claim 1, wherein said preloading is performed by a data storage controller connected to said boot device.	Burrows, as evidenced by the example citations below, discloses "said preloading is performed by a data storage controller connected to said boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said preloading is performed by a data storage controller connected to said boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Burrows discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> "These problems suggest that the compressor should be placed in the disk controller, but doing so requires more changes to LFS, and quite specialized hardware."

Burrows, 12.

> "But it is not possible to tell whether a particular disk block will fit in the current segment until compression has taken place, so we cannot determine what data should be written until the data has been transferred to the controller for compression."

Burrows, 12-13.

> "Once forward references are eliminated, our problem can be solved by placing a large buffer in the disk controller to hold the compressed data."

Burrows, 13.

> "Thus, the file system can prepare a larger amount of data for writing than will actually fit in the current segment, and can instruct the controller to truncate the write if it occupies more than the amount of physical space available."

Burrows, 13.

> "Finally, a disk controller containing a compressor must inform the software where each compressed block fell on the disk. Ideally, it would also construct the logical block map and append it to the data being written, in order to avoid an extra disk transfer to place the map at the end of the segment."

Burrows, 13.

> "The design can be adapted to use hardware compression devices, either combined with a disk controller or packaged separately."

Burrows, 18.

6. The method of claim 1, further comprising updating the list of boot data.	Burrows, as evidenced by the example citations below, discloses "updating the list of boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

Appendix B27
Invalidity of U.S. Patent 8,090,936 based on Burrows

8. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form.	Burrows, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above. *See also* > "Besides the algorithms based on Wheeler's ideas, we tried the LZRW1-A and LZRW3-A algorithms due to Williams. A full description and implementation are available elsewhere [15, 16], so we omit the details here." Burrows, 11. *See also* Table 2. > "For comparison, we include figures for the popular compress utility, which uses the LZC algorithm, and the zoo archiver, which uses the LZSS algorithm. The table shows that the algorithms we chose are quite fast, but better compression could be obtained by sacrificing speed. > Bell, Witten, and Cleary give a more thorough comparison of compression algorithms and their effectiveness on different sorts of data [3]. They also describe the LZC and LZSS algorithms." Burrows, 11-12.	

9. The method of claim 1, wherein a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form.	Burrows, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

11.1. a processor;	Burrows, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claim 1.2 above.	

11.2. a memory; and	Burrows, as evidenced by the example citations below, discloses “a memory.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Burrows discloses this limitation: *See* Claims 1.3, and 1.4 above.	

11.3.1 a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device,	Burrows, as evidenced by the example citations below, discloses “a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Burrows discloses this limitation: *See* Claims 1.1, 1.3, 1.4 above.	

11.3.2 said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system; and,	Burrows, as evidenced by the example citations below, discloses "said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claims 1.3, 1.4, and 1.5 above.	

11.4 a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data.	Burrows, as evidenced by the example citations below, discloses “a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Burrows discloses this limitation: *See* Claims 1.1 and 1.5 above.	

12. The system of claim 11, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program.	Burrows, as evidenced by the example citations below, discloses "said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claims 1.1, 3, and 11.3.1 above.	

13. The system of claim 11, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system.	Burrows, as evidenced by the example citations below, discloses "wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claims 1.1, 1.3, 3, 5 and 11.3.1 and 11.3.2 above.	

15. The system of claim 11, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form.	Burrows, as evidenced by the example citations below, discloses “Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Burrows discloses this limitation: *See* Claims 1, 8, and 11 above.	

16. The system of claim 11, wherein a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form.	Burrows, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Burrows discloses this limitation: *See* Claims 1, 9, and 11 above.	

Appendix B28
Invalidity of U.S. Patent 8,090,936 based on Cheng

The publication Cheng et al., Fast and highly reliable IBMLZ1 compression chip and algorithm for storage, Hot Chips V11, August 1995 ("Cheng") invalidates claims 1-6, 8-9, 11-13, and 15-16 of United States Patent No. 8,090,936 ("the '936 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '936 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

1.1 maintaining a list of boot data used for booting a computer system, wherein at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device;	Cheng, as evidenced by the exemplary citations below, discloses "maintaining a list of boot data used for booting a computer system, wherein at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system, wherein at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Cheng discloses this limitation:

> "Compression Benefits for Storage System
>
> • Compression x Compaction = 4.5 X - 6.0 X
>
> •System Performance"

Cheng, 144. *See also* Cheng, 144 Fig.1, Fig. 2.

> "Data compression allows more efficient use of storage media and communication bandwidth. Standard compression offerings for tape storage have been well-established since the late 1980s."

Cheng, 155.

> "Compression allows existing platforms and applications to benefit from a lower storage cost and potentially higher performance without any change to system hardware or software."

Cheng, 155.

> "The IBMLZl algorithm and technology was designed for high compression/decompression throughput with efficient hardware implementation, high reliability, low system overhead, and robust compression. Data integrity and reliability are ensured by coupled

compression-decompression checking, scrubbing operation, and extensive build-in checkings. Extremely low CPB=l^6 compression and decompression have been achieved. The extremely high compressing/decompressing throughput of 30 MB/sec-50 MB/sec allows transparent mode of operation and hence achieved minimal system overhead. The IBMLZl algorithm compresses well over the VM, MVS, RS6000, and PC test cases.”   Cheng, 163.

1.2 initializing a central processing unit of said computer system;	Cheng, as evidenced by the example citations below, discloses "initializing a central processing unit of said computer system;"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions..	

1.3 preloading said at least a portion of said boot data in compressed form from said boot device to a memory;	Cheng, as evidenced by the example citations below, discloses "preloading said at least a portion of said boot data in compressed form from said boot device to a memory;"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading said at least a portion of said boot data in compressed form from said boot device to a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

1.4 accessing and decompressing said at least a portion of said boot data in said compressed form from said memory; and	Cheng, as evidenced by the example citations below, discloses "accessing and decompressing said at least a portion of said boot data in said compressed form from said memory; and"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing and decompressing said at least a portion of said boot data in said compressed form from said memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

1.5 utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine.	Cheng, as evidenced by the example citations below, discloses "utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

2. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system.	Cheng, as evidenced by the example citations below, discloses "said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claims 1.1, 1.4, and 1.5 above.	

3. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system.	Cheng, as evidenced by the example citations below, discloses "said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Cheng discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> "The compression technology lowers the cost of storage without changes to any applications or data access methods."

Cheng, 155.

> "Compression allows existing platforms and applications to benefit from a lower storage cost and potentially higher performance without any change to system hardware or software."

Cheng, 155.

> "Robust Compression: achieve good coding efficiency for broad applications."

Cheng, 159.

4. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an application program and an operating system of said computer system.	Cheng, as evidenced by the example citations below, discloses "said decompressed at least a portion of said boot data comprises program code associated with an application program and an operating system of said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above. *See also* "The compression technology lowers the cost of storage without changes to any applications or data access methods." Cheng, 155. "Compression allows existing platforms and applications to benefit from a lower storage cost and potentially higher performance without any change to system hardware or software." Cheng, 155. "Robust Compression: achieve good coding efficiency for broad applications." Cheng, 159.	

5. The method of claim 1, wherein said preloading is performed by a data storage controller connected to said boot device.	Cheng, as evidenced by the example citations below, discloses "said preloading is performed by a data storage controller connected to said boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said preloading is performed by a data storage controller connected to said boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Cheng discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also ***<u>Look for additional references to controller</u>***

"DASO (DISK) Controller Compression and IBMLZ"

Cheng, 144.

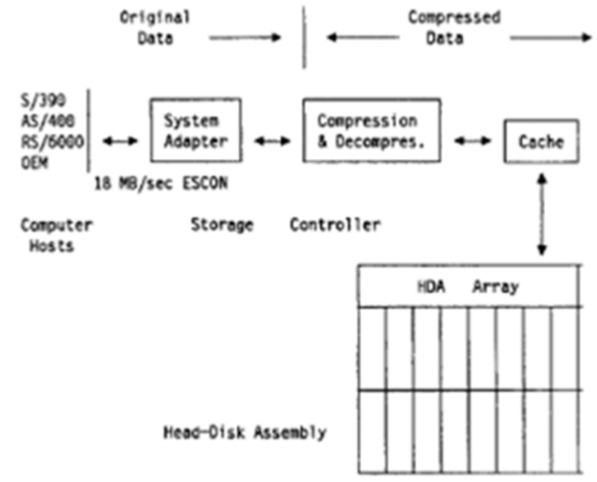


Fig. 1.

"Compression Objectives for Storage Controller"

Cheng, 144. *See also generally*, Cheng 144-145 (Compression Objectives for Storage Controller).

> "The IBMLZl is used in IBM's high-performance DASD controller family, tape drive family, and the AIX file system with compression."

Cheng, 160.

> "IBMLZl Compression Technology for Storage Controller"

Cheng, 163. *See also generally*, Cheng, 163 (IBMLZ1 Compression Technology for Storage Controller).

6. The method of claim 1, further comprising updating the list of boot data.	Cheng, as evidenced by the example citations below, discloses "updating the list of boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

Appendix B28
Invalidity of U.S. Patent 8,090,936 based on Cheng

8. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form.	Cheng, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Cheng discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> "DASO (DISK) Controller Compression and IBMLZ23
>
> •Lossless and General-Purpose Compression
>
> •Simple Algorithm Format and Robust
>
> • Optimized for High-Thruput Hardware Execution: 1 Byte/Cycle,
>
> 40MH/sec.
>
> • Algorithm Accepted as QIC 154 Standard"

Cheng, 144.

> "IBMLZ1 Algorithm and Hardware Development Paradigm

Extremely High Reliability	One undetected Error in 1030
Efficiencies	Robust Compression, Speed, Latency
System Requirement	Intra-Parallel instead Inter-Parallel, good Compression, Speed

Silicon Technology Regularity" Cheng, 146. *See also generally*, Cheng, 147-154. "The IBMLZ1 compression algorithm was designed not only for robust and highly efficient compression, but also for extremely high reliability. As compression removes redundancy in the source, the compressed data becomes extremely vulnerable to data corruption. Key design objectives for the IBMLZ1 development were: efficient hardware execution and efficient use of silicon technology; and minimum system integration overhead. Through new observations of pattern matching match-length distribution and use of graph vertex coloring for evaluating data flows, the IBMLZ1 compression algorithm and technology achieved the above objectives." Cheng, 155. "The LZl and the LZ2 compression algorithms are commonly regarded as Lempel and Ziv's compression algorithm 1 [ZL 77] and algorithm 2 [ZL 78] . The LZl and the LZ2, in their original form, expressed the notion of coding Model and bounds on compression. Professor Lempel noted that more than 90 percent of the compression software in the PC world is derived from either LZl or LZ2 class algorithms." Cheng, 157. *See also generally*, Cheng, 157-165.

9. The method of claim 1, wherein a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form.	Cheng, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Cheng discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> "Common methods used for redundancy reduction are run-length encoding, pattern matching, transformation, transform coding, and so on."

Cheng, 156.

> "The Huffman code, however, remains as the most popular encoding method for its simplicity and its effectiveness in general."

Cheng, 156. *See also generally*, Cheng, 156-165, Fig. 6.

11.1. a processor;	Cheng, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claim 1.2 above.	

11.2. a memory; and	Cheng, as evidenced by the example citations below, discloses “a memory.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Cheng discloses this limitation: *See* Claims 1.3, and 1.4 above.	

11.3.1 a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device,	Cheng, as evidenced by the example citations below, discloses "a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claims 1.1, 1.3, 1.4 above.	

11.3.2 said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system; and,	Cheng, as evidenced by the example citations below, discloses "said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claims 1.3, 1.4, and 1.5 above.	

11.4 a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data.	Cheng, as evidenced by the example citations below, discloses "a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claims 1.1 and 1.5 above.	

12. The system of claim 11, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program.	Cheng, as evidenced by the example citations below, discloses "said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claims 1.1, 3, and 11.3.1 above.	

13. The system of claim 11, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system.	Cheng, as evidenced by the example citations below, discloses "wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claims 1.1, 1.3, 3, 5 and 11.3.1 and 11.3.2 above.	

15. The system of claim 11, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form.	Cheng, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claims 1, 8, and 11 above.	

16. The system of claim 11, wherein a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form.	Cheng, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Cheng discloses this limitation: *See* Claims 1, 9, and 11 above.	

Appendix B29
Invalidity of U.S. Patent 8,090,936 based on Craft

The publication Craft, A Fast hardware data compression algorithm and some algorithmic extension, IBM J. Res. Develop., Vol 43, Nov. 1998, ("Craft") invalidates claims 1-6, 8-9, 11-13, and 15-16 of United States Patent No. 8,090,936 ("the '936 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '936 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix B29
Invalidity of U.S. Patent 8,090,936 based on Craft

1.1 maintaining a list of boot data used for booting a computer system, wherein at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device;	Craft, as evidenced by the exemplary citations below, discloses "maintaining a list of boot data used for booting a computer system, wherein at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system, wherein at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Craft discloses this limitation:

> "This paper reports on work at IBM's Austin and Burlington laboratories concerning fast hardware implementations of general-purpose lossless data compression algorithms, particularly for use in enhancing the data capacity of computer storage devices or systems, and transmission data rates for networking or telecommunications channels"

Craft, 733.

> "The scope of the work quickly expanded, however, as it became apparent that data compression technology could have tremendous implications for some major IBM business segments. In particular, its deployment within computer systems to enhance DASD storage capacity, or to increase the effective bandwidth of networking data channels, would present a major competitive advantage."

Craft, 734.

> "The encoding format for ALDC is presented, together with details of IBM's current fast hardware CMOS compression engine designs, based on use of a content addressable memory (CAM) array."

Craft, 733.

> "The initial IBM ALDC chips used entirely separate decompression and compression engines [5] and were in fact configured so that both could be operated simultaneously. However, in our later designs, we chose to add a conventional address decoder to the CAM so that it can also function as an SRAM during an LZ1 decode function. This results in a very compact hardware encoder/decoder, which we call a CRAM design, as it combines a compression engine CAM and a decompression engine RAM into one silicon array."

Craft, 738.

> "For the few customers requiring simultaneous compression/decompression capability, we can therefore put two such CRAM engines on the same chip, and the silicon area needed is still modest."

Craft, 738.

> "The hardware to implement both pre/postprocessors is trivial, requiring less than 10% of the CMOS chip area of the ALDC CRAM engine itself."

Craft, 744.

> "Using the more advanced IBM CMOS 6 and CMOS 7 technologies, we can easily fit multiple engines onto a single chip. In turn, this allows us to design ALDC compression systems which have sustained throughput in the gigabyte-per-second range, if required, and should effectively meet system storage and networking application requirements into the next millennium."

Craft, 744.

1.2 initializing a central processing unit of said computer system;	Craft, as evidenced by the example citations below, discloses "initializing a central processing unit of said computer system;"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

1.3 preloading said at least a portion of said boot data in compressed form from said boot device to a memory;	Craft, as evidenced by the example citations below, discloses "preloading said at least a portion of said boot data in compressed form from said boot device to a memory;"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading said at least a portion of said boot data in compressed form from said boot device to a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

1.4 accessing and decompressing said at least a portion of said boot data in said compressed form from said memory; and	Craft, as evidenced by the example citations below, discloses "accessing and decompressing said at least a portion of said boot data in said compressed form from said memory; and"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing and decompressing said at least a portion of said boot data in said compressed form from said memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

1.5 utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine.	Craft, as evidenced by the example citations below, discloses "utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

2. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system.	Craft, as evidenced by the example citations below, discloses "said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claims 1.1, 1.4, and 1.5 above.	

3. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system.	Craft, as evidenced by the example citations below, discloses "said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Craft discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> "It not only shows better compression on the smaller data block sizes that are desirable for random-access applications, it is also particularly amenable to a fast and simple CMOS hardware implementation based on the use of a content-addressable memory (CAM) array."

Craft, 734.

> "One cLDC preprocessor recodes runs of identical data bytes, and this is followed by one designed to recode the Unicode format, an increasingly important text data coding standard used by Java** and other Internet-based applications. ALDC compresses Unicode versions of ASCII text files some 40% worse than the original ASCII, but cLDC is able to remove this penalty completely."

Craft, 735.

See also generally, Craft 735-736 (Algorithms for DASD or networking applications).

4. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an application program and an operating system of said computer system.	Craft, as evidenced by the example citations below, discloses "said decompressed at least a portion of said boot data comprises program code associated with an application program and an operating system of said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above. *See also* "It not only shows better compression on the smaller data block sizes that are desirable for random-access applications, it is also particularly amenable to a fast and simple CMOS hardware implementation based on the use of a content-addressable memory (CAM) array." Craft, 734. "One cLDC preprocessor recodes runs of identical data bytes, and this is followed by one designed to recode the Unicode format, an increasingly important text data coding standard used by Java** and other Internet-based applications. ALDC compresses Unicode versions of ASCII text files some 40% worse than the original ASCII, but cLDC is able to remove this penalty completely." Craft, 735. *See also generally*, Craft 735-736 (Algorithms for DASD or networking applications).	

5. The method of claim 1, wherein said preloading is performed by a data storage controller connected to said boot device.	Craft, as evidenced by the example citations below, discloses “said preloading is performed by a data storage controller connected to said boot device.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said preloading is performed by a data storage controller connected to said boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Craft discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

6. The method of claim 1, further comprising updating the list of boot data.	Craft, as evidenced by the example citations below, discloses "updating the list of boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

8. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form.	Craft, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Craft discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> "Then, two main classes of adaptive Lempel-Ziv algorithm, now known as LZ1 and LZ2, are introduced. An outline of early work comparing these two types of algorithm is presented, together with some fundamental distinctions which led to the choice and development of an IBM variant of the LZ1 algorithm, ALDC, and its implementation in hardware."

Craft, 733.

> "Adaptive data compression techniques try to construct models, or look for data sequences derived in some fashion from recent experience. The algorithm thus adapts dynamically to different types of data. There are two classes of adaptive algorithm which are generally acknowledged to be among the most effective, yielding good compression over a wide range of data types. These were both first proposed by A. Lempel and J. Ziv, in 1977 and 1978, and are commonly now referred to as LZ1 and LZ2 respectively [1-3].

Craft, 734. See also generally, Craft 734-744.

> "This algorithm employs two cascaded pre/postprocessors, one designed to recode a run of identical byte values, the other to detect and recode Unicode-like data sequences. Unicode [13] is used by the Java language and other Web-based applications."

Craft, 742. "The CRAM compression technology is clearly able to cover an extremely wide range of applicability. Its small size allows integration into the smallest portable, handheld, or wireless applications, where it is much faster and consumes far less power than software." Craft, 744.

Appendix B29
Invalidity of U.S. Patent 8,090,936 based on Craft

9. The method of claim 1, wherein a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form.	Craft, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Craft discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> "However, in our later designs, we chose to add a conventional address decoder to the CAM so that it can also function as an SRAM during an LZ1 decode function. This results in a very compact hardware encoder/decoder, which we call a CRAM design, as it combines a compression engine CAM and a decompression engine RAM into one silicon array."

Craft, 738.

> "An LZ1 decompressor builds and maintains an identical history copy, which it updates in the same manner as the encoder, as each LITERAL or COPY_POINTER is processed."

Craft, 738.

> "The code byte-stream output from this preprocessor is then fed into a standard ALDC-1 encoder. For decompression, an ALDC decoder generates code byte values which are then fed into a hardware postprocessor to reconstruct the original data bit stream."

Craft, 741.

11.1. a processor;	Craft, as evidenced by the example citations below, discloses “a processor.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Craft discloses this limitation: *See* Claim 1.2 above.	

11.2. a memory; and	Craft, as evidenced by the example citations below, discloses “a memory.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Craft discloses this limitation: *See* Claims 1.3, and 1.4 above.	

11.3.1 a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device,	Craft, as evidenced by the example citations below, discloses "a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claims 1.1, 1.3, 1.4 above.	

11.3.2 said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system; and,	Craft, as evidenced by the example citations below, discloses “said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Craft discloses this limitation: *See* Claims 1.3, 1.4, and 1.5 above.	

11.4 a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data.	Craft, as evidenced by the example citations below, discloses "a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claims 1.1 and 1.5 above.	

Appendix B29
Invalidity of U.S. Patent 8,090,936 based on Craft

12. The system of claim 11, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program.	Craft, as evidenced by the example citations below, discloses "said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claims 1.1, 3, and 11.3.1 above.	

13. The system of claim 11, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system.	Craft, as evidenced by the example citations below, discloses "wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claims 1.1, 1.3, 3, 5 and 11.3.1 and 11.3.2 above.	

15. The system of claim 11, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form.	Craft, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Craft discloses this limitation: *See* Claims 1, 8, and 11 above.	

16. The system of claim 11, wherein a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form.	Craft, as evidenced by the example citations below, discloses “a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Craft discloses this limitation: *See* Claims 1, 9, and 11 above.	

Appendix B30
Invalidity of U.S. Patent 8,090,936 based on Douglis

Douglis, "One the Role of Compression in Distributed Systems" ("Douglis 1") and Douglis, "The Compression Cache: Using On-Line Compression to Extend Physical Memory," Winter 1993 USENIX Conference, Jan 1993 ("Douglis 2") (collectively, "Douglis"), alone or in combination, invalidate claims 1-6, 8-9, 11-13, and 15-16 of United States Patent No. 8,090,936 ("the '936 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

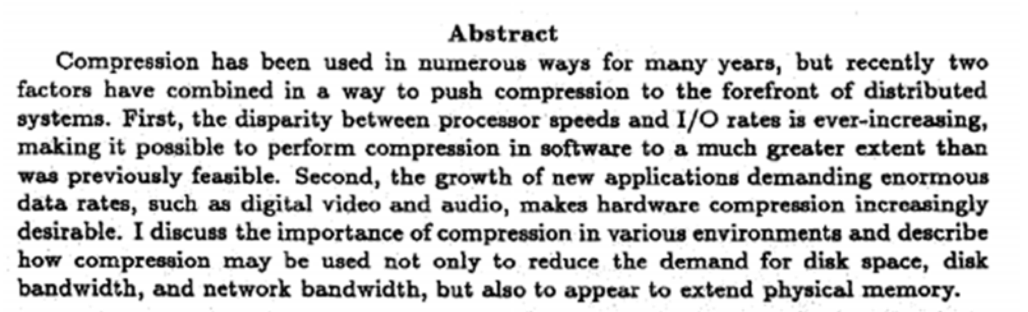
The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '936 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix B30
Invalidity of U.S. Patent 8,090,936 based on Douglis

1.1 maintaining a list of boot data used for booting a computer system, wherein at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device;	Douglis, as evidenced by the exemplary citations below, discloses "maintaining a list of boot data used for booting a computer system, wherein at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system, wherein at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

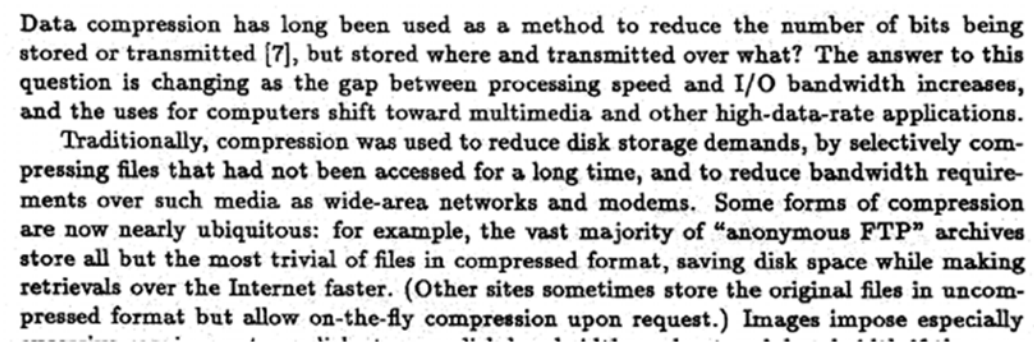
Douglis discloses this limitation:



Abstract

Compression has been used in numerous ways for many years, but recently two factors have combined in a way to push compression to the forefront of distributed systems. First, the disparity between processor speeds and I/O rates is ever-increasing, making it possible to perform compression in software to a much greater extent than was previously feasible. Second, the growth of new applications demanding enormous data rates, such as digital video and audio, makes hardware compression increasingly desirable. I discuss the importance of compression in various environments and describe how compression may be used not only to reduce the demand for disk space, disk bandwidth, and network bandwidth, but also to appear to extend physical memory.

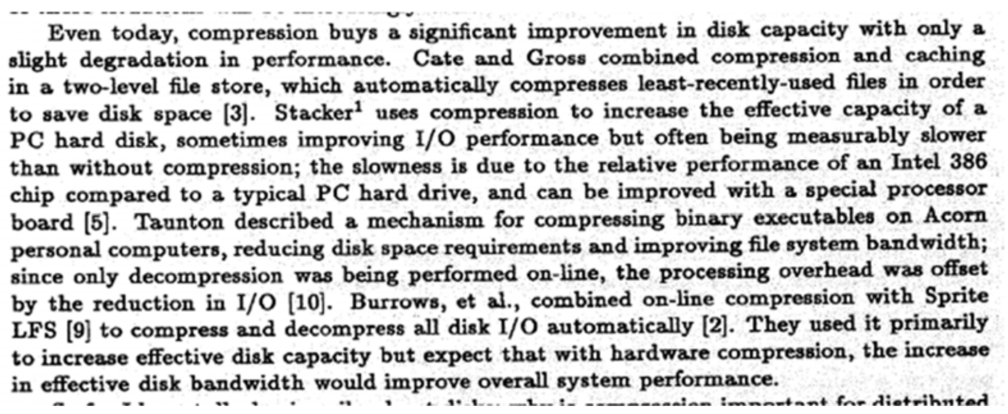
Douglis 1, Abstact



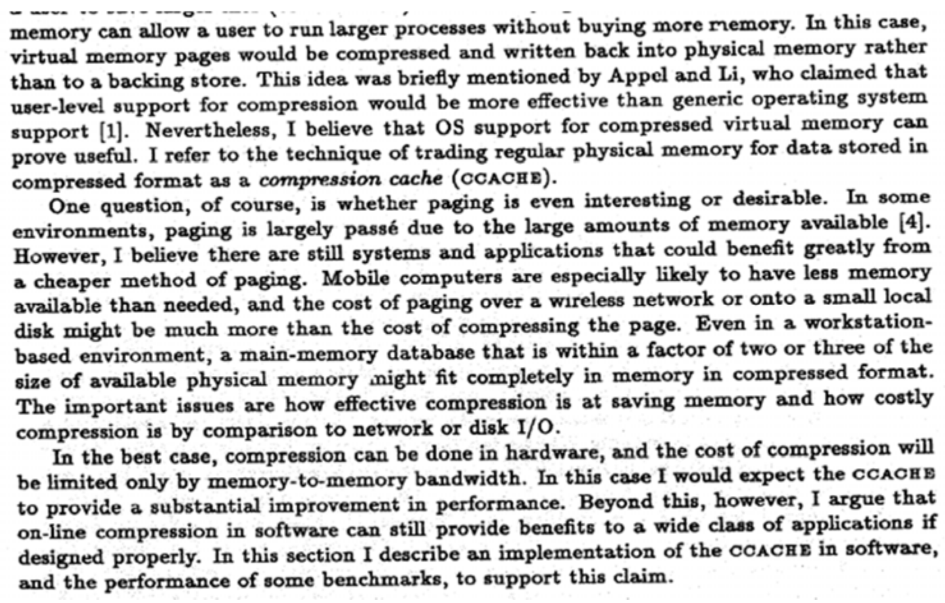
Data compression has long been used as a method to reduce the number of bits being stored or transmitted [7], but stored where and transmitted over what? The answer to this question is changing as the gap between processing speed and I/O bandwidth increases, and the uses for computers shift toward multimedia and other high-data-rate applications.

Traditionally, compression was used to reduce disk storage demands, by selectively compressing files that had not been accessed for a long time, and to reduce bandwidth requirements over such media as wide-area networks and modems. Some forms of compression are now nearly ubiquitous: for example, the vast majority of "anonymous FTP" archives store all but the most trivial of files in compressed format, saving disk space while making retrievals over the Internet faster. (Other sites sometimes store the original files in uncompressed format but allow on-the-fly compression upon request.) Images impose especially

Douglis 1, at 1


Even today, compression buys a significant improvement in disk capacity with only a slight degradation in performance. Cate and Gross combined compression and caching in a two-level file store, which automatically compresses least-recently-used files in order to save disk space [3]. Stacker[1] uses compression to increase the effective capacity of a PC hard disk, sometimes improving I/O performance but often being measurably slower than without compression; the slowness is due to the relative performance of an Intel 386 chip compared to a typical PC hard drive, and can be improved with a special processor board [5]. Taunton described a mechanism for compressing binary executables on Acorn personal computers, reducing disk space requirements and improving file system bandwidth; since only decompression was being performed on-line, the processing overhead was offset by the reduction in I/O [10]. Burrows, et al., combined on-line compression with Sprite LFS [9] to compress and decompress all disk I/O automatically [2]. They used it primarily to increase effective disk capacity but expect that with hardware compression, the increase in effective disk bandwidth would improve overall system performance.

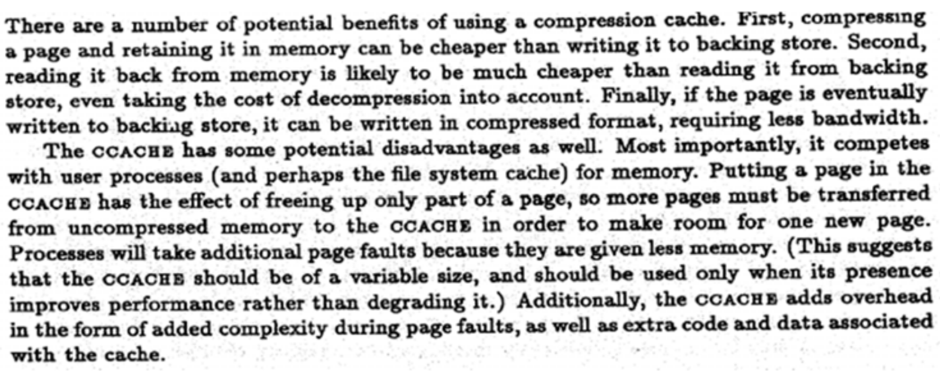
Douglis 1, 2


memory can allow a user to run larger processes without buying more memory. In this case, virtual memory pages would be compressed and written back into physical memory rather than to a backing store. This idea was briefly mentioned by Appel and Li, who claimed that user-level support for compression would be more effective than generic operating system support [1]. Nevertheless, I believe that OS support for compressed virtual memory can prove useful. I refer to the technique of trading regular physical memory for data stored in compressed format as a *compression cache* (CCACHE).

One question, of course, is whether paging is even interesting or desirable. In some environments, paging is largely passé due to the large amounts of memory available [4]. However, I believe there are still systems and applications that could benefit greatly from a cheaper method of paging. Mobile computers are especially likely to have less memory available than needed, and the cost of paging over a wireless network or onto a small local disk might be much more than the cost of compressing the page. Even in a workstation-based environment, a main-memory database that is within a factor of two or three of the size of available physical memory might fit completely in memory in compressed format. The important issues are how effective compression is at saving memory and how costly compression is by comparison to network or disk I/O.

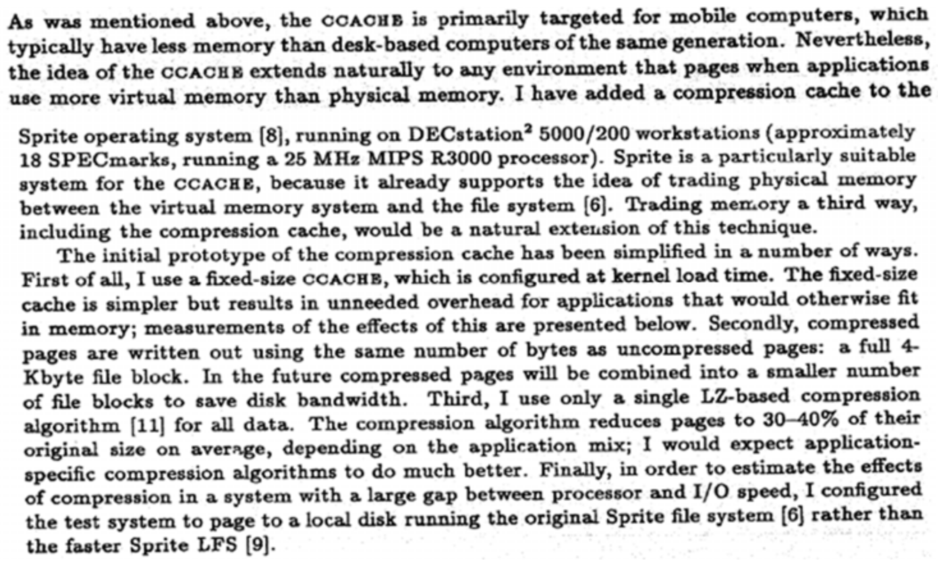
In the best case, compression can be done in hardware, and the cost of compression will be limited only by memory-to-memory bandwidth. In this case I would expect the CCACHE to provide a substantial improvement in performance. Beyond this, however, I argue that on-line compression in software can still provide benefits to a wide class of applications if designed properly. In this section I describe an implementation of the CCACHE in software, and the performance of some benchmarks, to support this claim.

Douglis 1, 2


There are a number of potential benefits of using a compression cache. First, compressing a page and retaining it in memory can be cheaper than writing it to backing store. Second, reading it back from memory is likely to be much cheaper than reading it from backing store, even taking the cost of decompression into account. Finally, if the page is eventually written to backing store, it can be written in compressed format, requiring less bandwidth.

The CCACHE has some potential disadvantages as well. Most importantly, it competes with user processes (and perhaps the file system cache) for memory. Putting a page in the CCACHE has the effect of freeing up only part of a page, so more pages must be transferred from uncompressed memory to the CCACHE in order to make room for one new page. Processes will take additional page faults because they are given less memory. (This suggests that the CCACHE should be of a variable size, and should be used only when its presence improves performance rather than degrading it.) Additionally, the CCACHE adds overhead in the form of added complexity during page faults, as well as extra code and data associated with the cache.

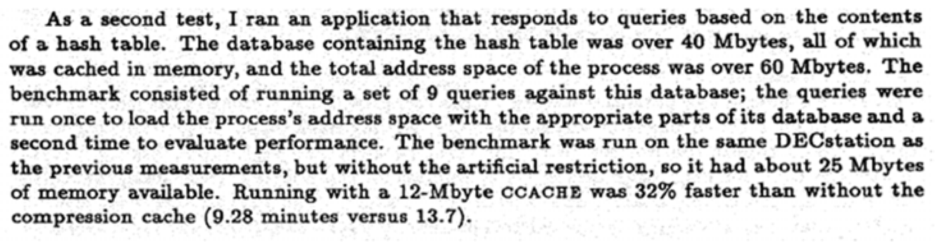
Douglis 1, 3.1


As was mentioned above, the CCACHE is primarily targeted for mobile computers, which typically have less memory than desk-based computers of the same generation. Nevertheless, the idea of the CCACHE extends naturally to any environment that pages when applications use more virtual memory than physical memory. I have added a compression cache to the

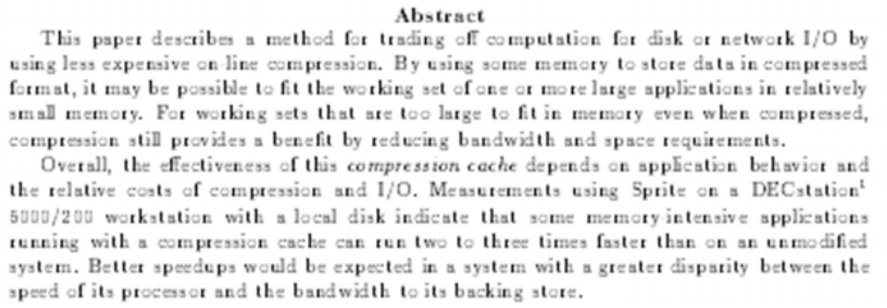
Sprite operating system [8], running on DECstation[2] 5000/200 workstations (approximately 18 SPECmarks, running a 25 MHz MIPS R3000 processor). Sprite is a particularly suitable system for the CCACHE, because it already supports the idea of trading physical memory between the virtual memory system and the file system [6]. Trading memory a third way, including the compression cache, would be a natural extension of this technique.

The initial prototype of the compression cache has been simplified in a number of ways. First of all, I use a fixed-size CCACHE, which is configured at kernel load time. The fixed-size cache is simpler but results in unneeded overhead for applications that would otherwise fit in memory; measurements of the effects of this are presented below. Secondly, compressed pages are written out using the same number of bytes as uncompressed pages: a full 4-Kbyte file block. In the future compressed pages will be combined into a smaller number of file blocks to save disk bandwidth. Third, I use only a single LZ-based compression algorithm [11] for all data. The compression algorithm reduces pages to 30–40% of their original size on average, depending on the application mix; I would expect application-specific compression algorithms to do much better. Finally, in order to estimate the effects of compression in a system with a large gap between processor and I/O speed, I configured the test system to page to a local disk running the original Sprite file system [6] rather than the faster Sprite LFS [9].

Douglis 1, 3.2


As a second test, I ran an application that responds to queries based on the contents of a hash table. The database containing the hash table was over 40 Mbytes, all of which was cached in memory, and the total address space of the process was over 60 Mbytes. The benchmark consisted of running a set of 9 queries against this database; the queries were run once to load the process's address space with the appropriate parts of its database and a second time to evaluate performance. The benchmark was run on the same DECstation as the previous measurements, but without the artificial restriction, so it had about 25 Mbytes of memory available. Running with a 12-Mbyte CCACHE was 32% faster than without the compression cache (9.28 minutes versus 13.7).

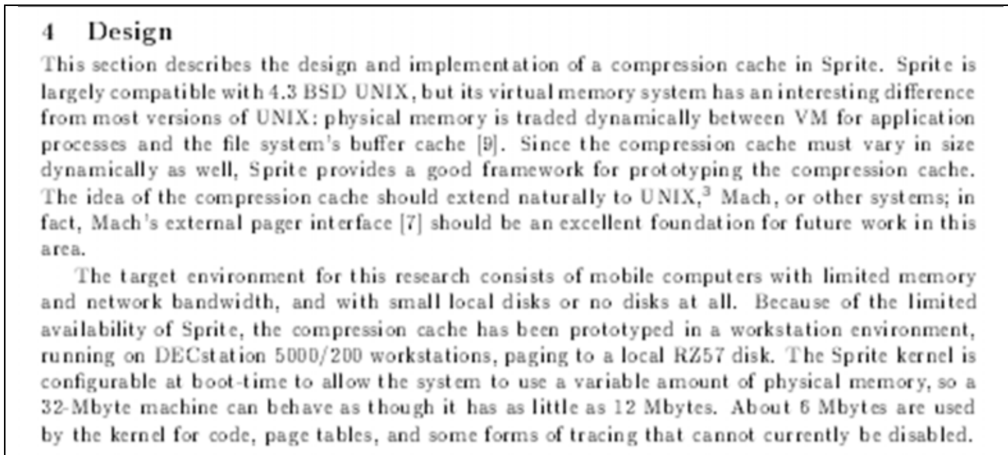
Douglis 1, 3.4


Abstract

This paper describes a method for trading off computation for disk or network I/O by using less expensive on-line compression. By using some memory to store data in compressed format, it may be possible to fit the working set of one or more large applications in relatively small memory. For working sets that are too large to fit in memory even when compressed, compression still provides a benefit by reducing bandwidth and space requirements.

Overall, the effectiveness of this *compression cache* depends on application behavior and the relative costs of compression and I/O. Measurements using Sprite on a DECstation[1] 5000/200 workstation with a local disk indicate that some memory-intensive applications running with a compression cache can run two to three times faster than on an unmodified system. Better speedups would be expected in a system with a greater disparity between the speed of its processor and the bandwidth to its backing store.

Douglis 2, Abstract



4 Design

This section describes the design and implementation of a compression cache in Sprite. Sprite is largely compatible with 4.3 BSD UNIX, but its virtual memory system has an interesting difference from most versions of UNIX: physical memory is traded dynamically between VM for application processes and the file system's buffer cache [9]. Since the compression cache must vary in size dynamically as well, Sprite provides a good framework for prototyping the compression cache. The idea of the compression cache should extend naturally to UNIX,[3] Mach, or other systems; in fact, Mach's external pager interface [7] should be an excellent foundation for future work in this area.

The target environment for this research consists of mobile computers with limited memory and network bandwidth, and with small local disks or no disks at all. Because of the limited availability of Sprite, the compression cache has been prototyped in a workstation environment, running on DECstation 5000/200 workstations, paging to a local RZ57 disk. The Sprite kernel is configurable at boot-time to allow the system to use a variable amount of physical memory, so a 32-Mbyte machine can behave as though it has as little as 12 Mbytes. About 6 Mbytes are used by the kernel for code, page tables, and some forms of tracing that cannot currently be disabled.

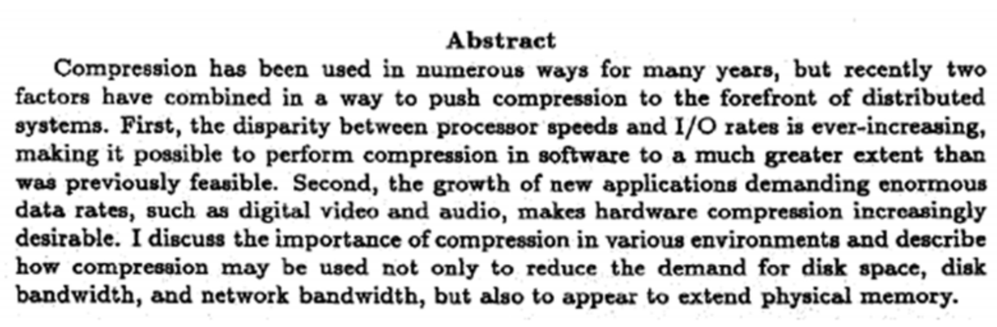
Douglis 2, 4

Appendix B30
Invalidity of U.S. Patent 8,090,936 based on Douglis

1.2 initializing a central processing unit of said computer system;	Douglis, as evidenced by the example citations below, discloses “initializing a central processing unit of said computer system;”

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.

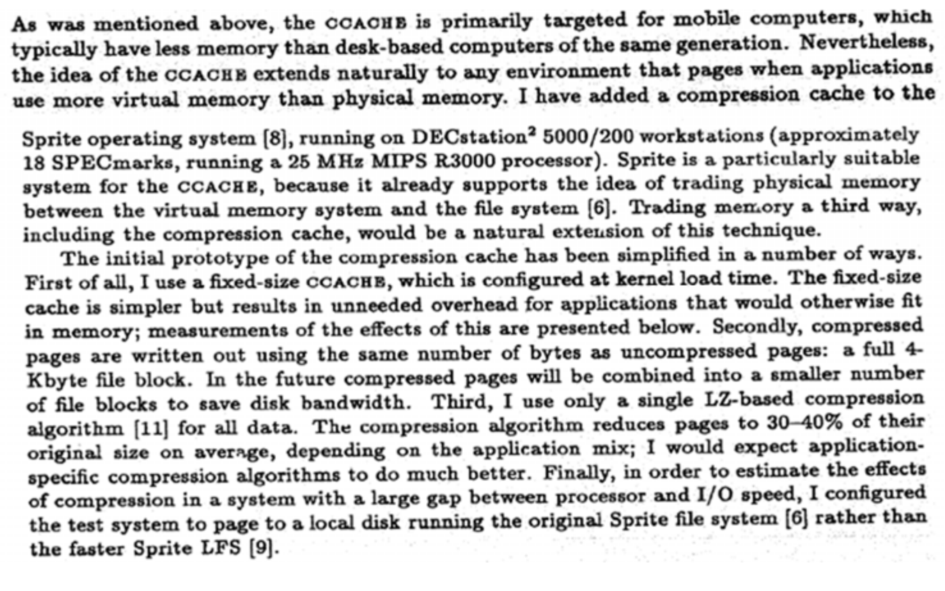
Douglis discloses this limitation:



Abstract

Compression has been used in numerous ways for many years, but recently two factors have combined in a way to push compression to the forefront of distributed systems. First, the disparity between processor speeds and I/O rates is ever-increasing, making it possible to perform compression in software to a much greater extent than was previously feasible. Second, the growth of new applications demanding enormous data rates, such as digital video and audio, makes hardware compression increasingly desirable. I discuss the importance of compression in various environments and describe how compression may be used not only to reduce the demand for disk space, disk bandwidth, and network bandwidth, but also to appear to extend physical memory.

Douglis 1, Abstact

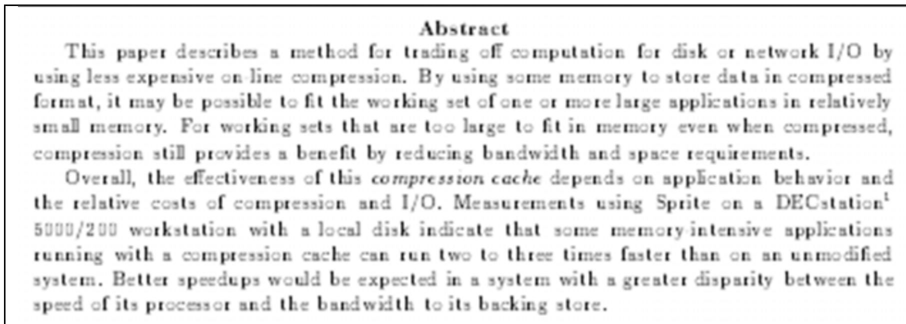


As was mentioned above, the CCACHE is primarily targeted for mobile computers, which typically have less memory than desk-based computers of the same generation. Nevertheless, the idea of the CCACHE extends naturally to any environment that pages when applications use more virtual memory than physical memory. I have added a compression cache to the

Sprite operating system [8], running on DECstation[2] 5000/200 workstations (approximately 18 SPECmarks, running a 25 MHz MIPS R3000 processor). Sprite is a particularly suitable system for the CCACHE, because it already supports the idea of trading physical memory between the virtual memory system and the file system [6]. Trading memory a third way, including the compression cache, would be a natural extension of this technique.

The initial prototype of the compression cache has been simplified in a number of ways. First of all, I use a fixed-size CCACHE, which is configured at kernel load time. The fixed-size cache is simpler but results in unneeded overhead for applications that would otherwise fit in memory; measurements of the effects of this are presented below. Secondly, compressed pages are written out using the same number of bytes as uncompressed pages: a full 4-Kbyte file block. In the future compressed pages will be combined into a smaller number of file blocks to save disk bandwidth. Third, I use only a single LZ-based compression algorithm [11] for all data. The compression algorithm reduces pages to 30–40% of their original size on average, depending on the application mix; I would expect application-specific compression algorithms to do much better. Finally, in order to estimate the effects of compression in a system with a large gap between processor and I/O speed, I configured the test system to page to a local disk running the original Sprite file system [6] rather than the faster Sprite LFS [9].

Douglis 1, 3.2

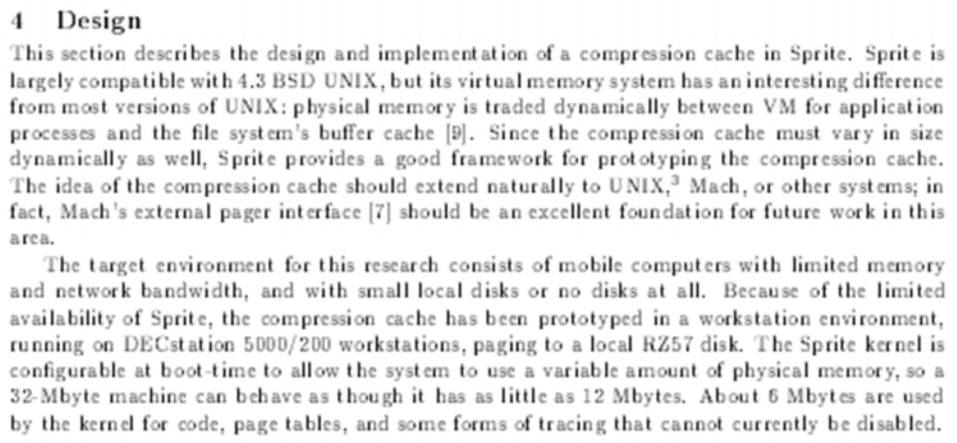


Abstract

This paper describes a method for trading off computation for disk or network I/O by using less expensive on-line compression. By using some memory to store data in compressed format, it may be possible to fit the working set of one or more large applications in relatively small memory. For working sets that are too large to fit in memory even when compressed, compression still provides a benefit by reducing bandwidth and space requirements.

Overall, the effectiveness of this *compression cache* depends on application behavior and the relative costs of compression and I/O. Measurements using Sprite on a DECstation[1] 5000/200 workstation with a local disk indicate that some memory-intensive applications running with a compression cache can run two to three times faster than on an unmodified system. Better speedups would be expected in a system with a greater disparity between the speed of its processor and the bandwidth to its backing store.

Douglis 2, Abstract



4 Design

This section describes the design and implementation of a compression cache in Sprite. Sprite is largely compatible with 4.3 BSD UNIX, but its virtual memory system has an interesting difference from most versions of UNIX: physical memory is traded dynamically between VM for application processes and the file system's buffer cache [9]. Since the compression cache must vary in size dynamically as well, Sprite provides a good framework for prototyping the compression cache. The idea of the compression cache should extend naturally to UNIX,[3] Mach, or other systems; in fact, Mach's external pager interface [7] should be an excellent foundation for future work in this area.

The target environment for this research consists of mobile computers with limited memory and network bandwidth, and with small local disks or no disks at all. Because of the limited availability of Sprite, the compression cache has been prototyped in a workstation environment, running on DECstation 5000/200 workstations, paging to a local RZ57 disk. The Sprite kernel is configurable at boot-time to allow the system to use a variable amount of physical memory, so a 32-Mbyte machine can behave as though it has as little as 12 Mbytes. About 6 Mbytes are used by the kernel for code, page tables, and some forms of tracing that cannot currently be disabled.

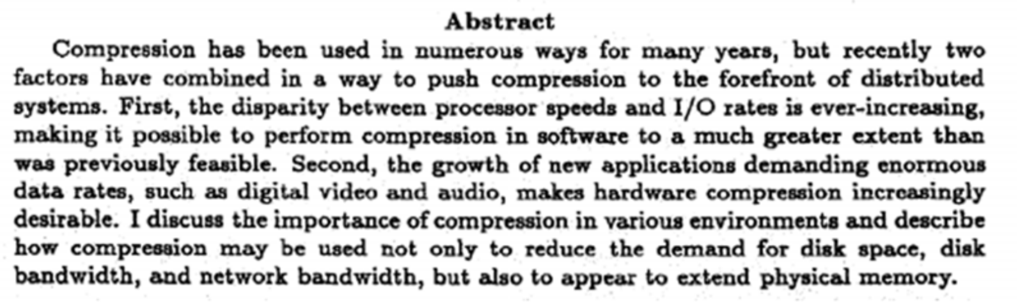
Douglis 2, 4

Appendix B30
Invalidity of U.S. Patent 8,090,936 based on Douglis

1.3 preloading said at least a portion of said boot data in compressed form from said boot device to a memory;	Douglis, as evidenced by the example citations below, discloses "preloading said at least a portion of said boot data in compressed form from said boot device to a memory;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading said at least a portion of said boot data in compressed form from said boot device to a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

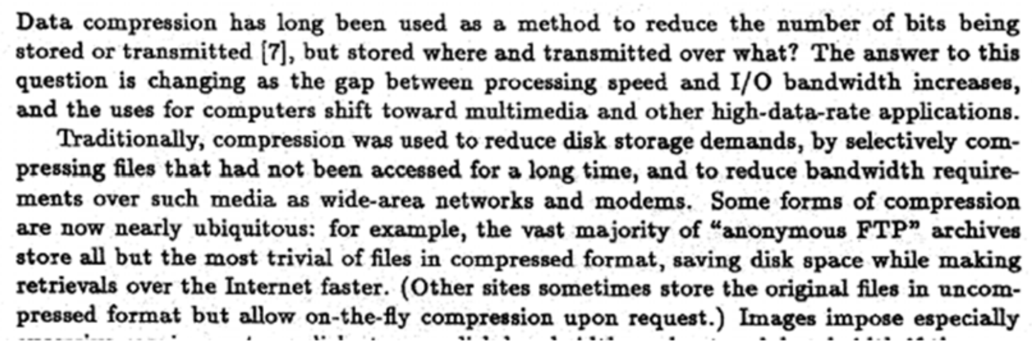
Douglis discloses this limitation:



Abstract

Compression has been used in numerous ways for many years, but recently two factors have combined in a way to push compression to the forefront of distributed systems. First, the disparity between processor speeds and I/O rates is ever-increasing, making it possible to perform compression in software to a much greater extent than was previously feasible. Second, the growth of new applications demanding enormous data rates, such as digital video and audio, makes hardware compression increasingly desirable. I discuss the importance of compression in various environments and describe how compression may be used not only to reduce the demand for disk space, disk bandwidth, and network bandwidth, but also to appear to extend physical memory.

Douglis 1, Abstact

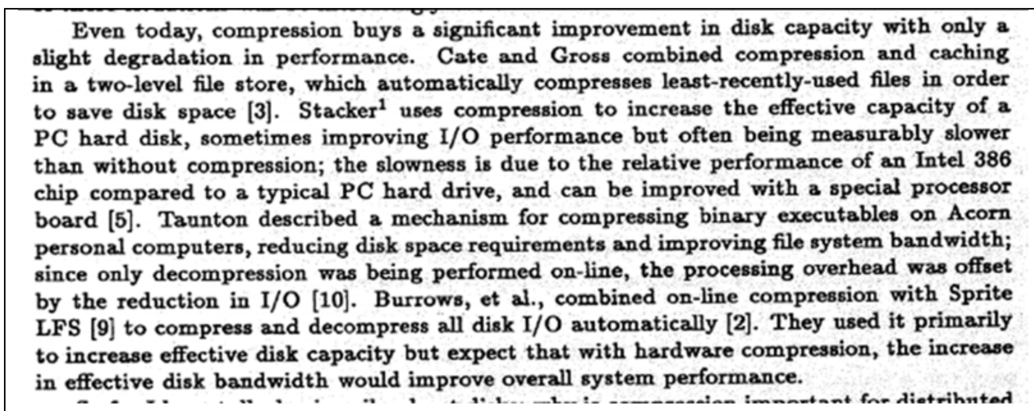


Data compression has long been used as a method to reduce the number of bits being stored or transmitted [7], but stored where and transmitted over what? The answer to this question is changing as the gap between processing speed and I/O bandwidth increases, and the uses for computers shift toward multimedia and other high-data-rate applications.

Traditionally, compression was used to reduce disk storage demands, by selectively compressing files that had not been accessed for a long time, and to reduce bandwidth requirements over such media as wide-area networks and modems. Some forms of compression are now nearly ubiquitous: for example, the vast majority of "anonymous FTP" archives store all but the most trivial of files in compressed format, saving disk space while making retrievals over the Internet faster. (Other sites sometimes store the original files in uncompressed format but allow on-the-fly compression upon request.) Images impose especially

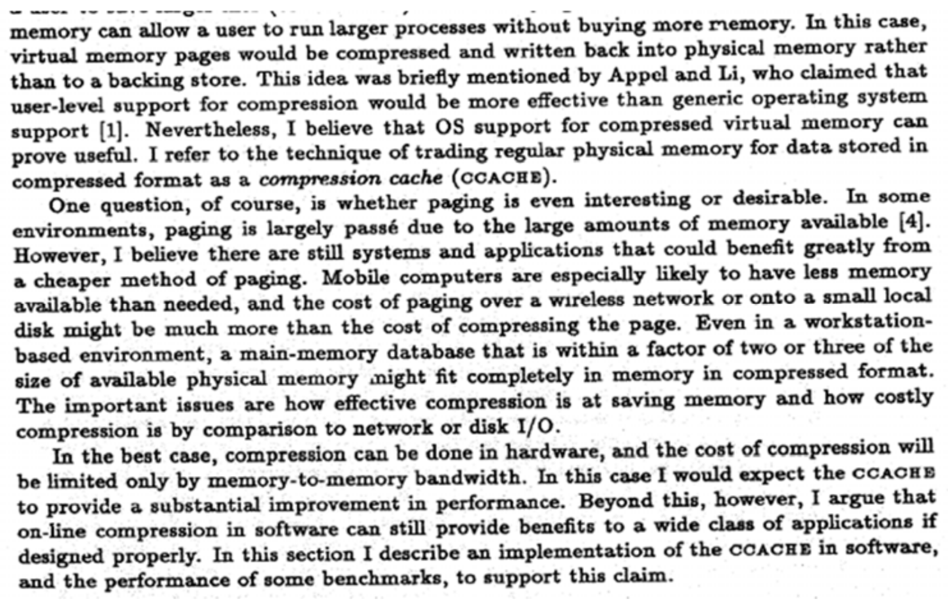
Douglis 1, at 1

Appendix B30
Invalidity of U.S. Patent 8,090,936 based on Douglis



Even today, compression buys a significant improvement in disk capacity with only a slight degradation in performance. Cate and Gross combined compression and caching in a two-level file store, which automatically compresses least-recently-used files in order to save disk space [3]. Stacker[1] uses compression to increase the effective capacity of a PC hard disk, sometimes improving I/O performance but often being measurably slower than without compression; the slowness is due to the relative performance of an Intel 386 chip compared to a typical PC hard drive, and can be improved with a special processor board [5]. Taunton described a mechanism for compressing binary executables on Acorn personal computers, reducing disk space requirements and improving file system bandwidth; since only decompression was being performed on-line, the processing overhead was offset by the reduction in I/O [10]. Burrows, et al., combined on-line compression with Sprite LFS [9] to compress and decompress all disk I/O automatically [2]. They used it primarily to increase effective disk capacity but expect that with hardware compression, the increase in effective disk bandwidth would improve overall system performance.

Douglis 1, 2

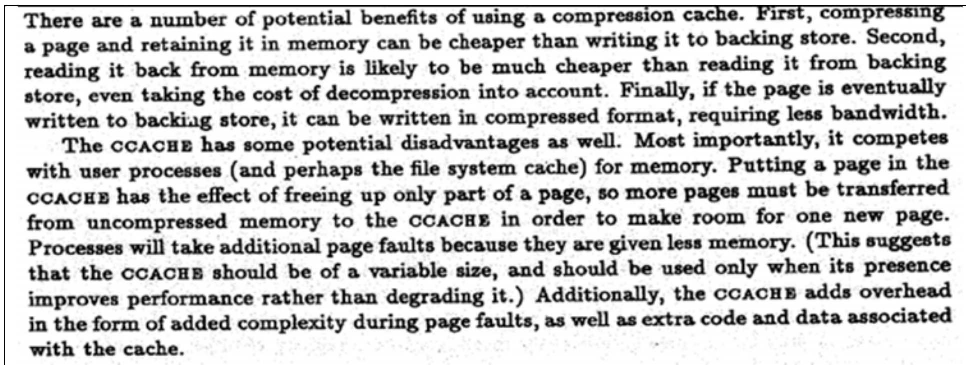


memory can allow a user to run larger processes without buying more memory. In this case, virtual memory pages would be compressed and written back into physical memory rather than to a backing store. This idea was briefly mentioned by Appel and Li, who claimed that user-level support for compression would be more effective than generic operating system support [1]. Nevertheless, I believe that OS support for compressed virtual memory can prove useful. I refer to the technique of trading regular physical memory for data stored in compressed format as a *compression cache* (CCACHE).

One question, of course, is whether paging is even interesting or desirable. In some environments, paging is largely passé due to the large amounts of memory available [4]. However, I believe there are still systems and applications that could benefit greatly from a cheaper method of paging. Mobile computers are especially likely to have less memory available than needed, and the cost of paging over a wireless network or onto a small local disk might be much more than the cost of compressing the page. Even in a workstation-based environment, a main-memory database that is within a factor of two or three of the size of available physical memory might fit completely in memory in compressed format. The important issues are how effective compression is at saving memory and how costly compression is by comparison to network or disk I/O.

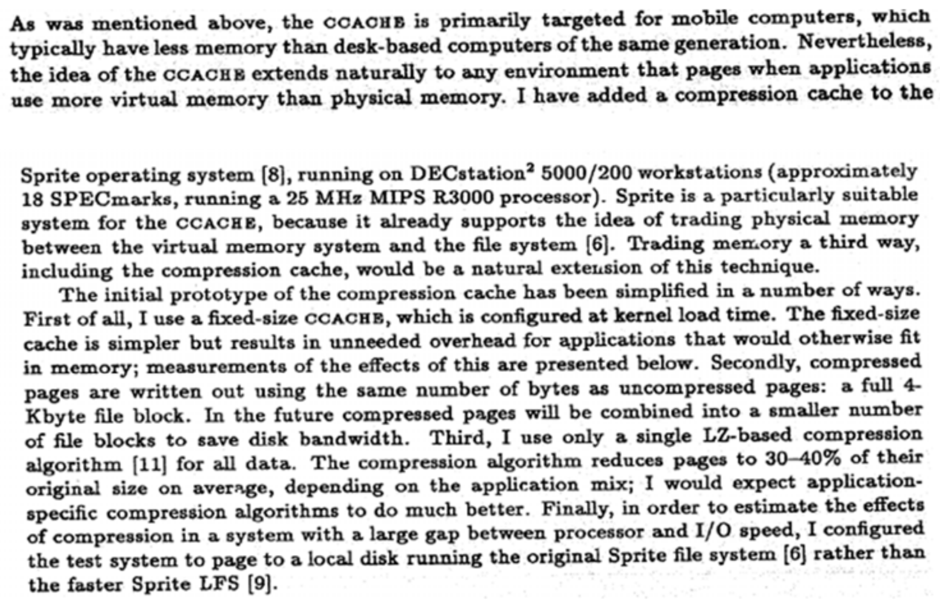
In the best case, compression can be done in hardware, and the cost of compression will be limited only by memory-to-memory bandwidth. In this case I would expect the CCACHE to provide a substantial improvement in performance. Beyond this, however, I argue that on-line compression in software can still provide benefits to a wide class of applications if designed properly. In this section I describe an implementation of the CCACHE in software, and the performance of some benchmarks, to support this claim.

Douglis 1, 2


There are a number of potential benefits of using a compression cache. First, compressing a page and retaining it in memory can be cheaper than writing it to backing store. Second, reading it back from memory is likely to be much cheaper than reading it from backing store, even taking the cost of decompression into account. Finally, if the page is eventually written to backing store, it can be written in compressed format, requiring less bandwidth.

The CCACHE has some potential disadvantages as well. Most importantly, it competes with user processes (and perhaps the file system cache) for memory. Putting a page in the CCACHE has the effect of freeing up only part of a page, so more pages must be transferred from uncompressed memory to the CCACHE in order to make room for one new page. Processes will take additional page faults because they are given less memory. (This suggests that the CCACHE should be of a variable size, and should be used only when its presence improves performance rather than degrading it.) Additionally, the CCACHE adds overhead in the form of added complexity during page faults, as well as extra code and data associated with the cache.

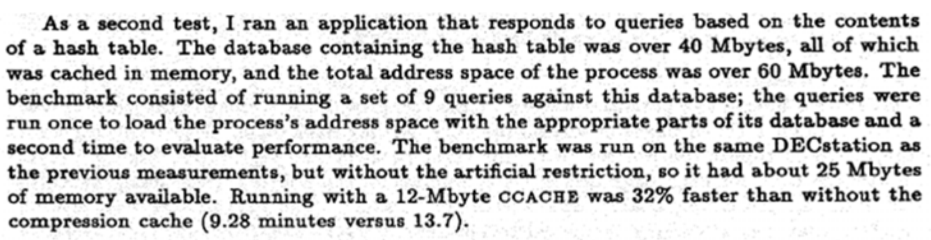
Douglis 1, 3.1


As was mentioned above, the CCACHE is primarily targeted for mobile computers, which typically have less memory than desk-based computers of the same generation. Nevertheless, the idea of the CCACHE extends naturally to any environment that pages when applications use more virtual memory than physical memory. I have added a compression cache to the

Sprite operating system [8], running on DECstation[2] 5000/200 workstations (approximately 18 SPECmarks, running a 25 MHz MIPS R3000 processor). Sprite is a particularly suitable system for the CCACHE, because it already supports the idea of trading physical memory between the virtual memory system and the file system [6]. Trading memory a third way, including the compression cache, would be a natural extension of this technique.

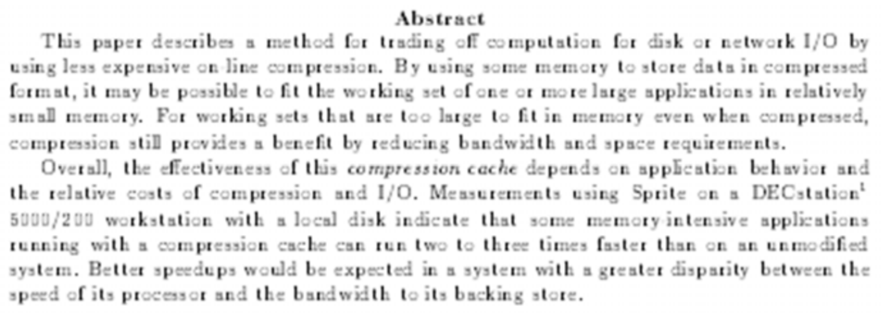
The initial prototype of the compression cache has been simplified in a number of ways. First of all, I use a fixed-size CCACHE, which is configured at kernel load time. The fixed-size cache is simpler but results in unneeded overhead for applications that would otherwise fit in memory; measurements of the effects of this are presented below. Secondly, compressed pages are written out using the same number of bytes as uncompressed pages: a full 4-Kbyte file block. In the future compressed pages will be combined into a smaller number of file blocks to save disk bandwidth. Third, I use only a single LZ-based compression algorithm [11] for all data. The compression algorithm reduces pages to 30–40% of their original size on average, depending on the application mix; I would expect application-specific compression algorithms to do much better. Finally, in order to estimate the effects of compression in a system with a large gap between processor and I/O speed, I configured the test system to page to a local disk running the original Sprite file system [6] rather than the faster Sprite LFS [9].

Douglis 1, 3.2


As a second test, I ran an application that responds to queries based on the contents of a hash table. The database containing the hash table was over 40 Mbytes, all of which was cached in memory, and the total address space of the process was over 60 Mbytes. The benchmark consisted of running a set of 9 queries against this database; the queries were run once to load the process's address space with the appropriate parts of its database and a second time to evaluate performance. The benchmark was run on the same DECstation as the previous measurements, but without the artificial restriction, so it had about 25 Mbytes of memory available. Running with a 12-Mbyte CCACHE was 32% faster than without the compression cache (9.28 minutes versus 13.7).

Douglis 1, 3.4

Appendix B30
Invalidity of U.S. Patent 8,090,936 based on Douglis

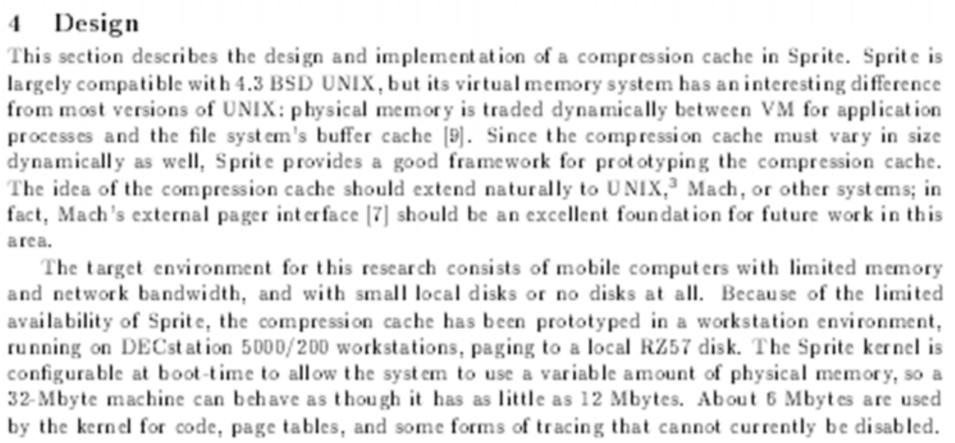


Abstract

This paper describes a method for trading off computation for disk or network I/O by using less expensive on-line compression. By using some memory to store data in compressed format, it may be possible to fit the working set of one or more large applications in relatively small memory. For working sets that are too large to fit in memory even when compressed, compression still provides a benefit by reducing bandwidth and space requirements.

Overall, the effectiveness of this *compression cache* depends on application behavior and the relative costs of compression and I/O. Measurements using Sprite on a DECstation[1] 5000/200 workstation with a local disk indicate that some memory-intensive applications running with a compression cache can run two to three times faster than on an unmodified system. Better speedups would be expected in a system with a greater disparity between the speed of its processor and the bandwidth to its backing store.

Douglis 2, Abstract



4 Design

This section describes the design and implementation of a compression cache in Sprite. Sprite is largely compatible with 4.3 BSD UNIX, but its virtual memory system has an interesting difference from most versions of UNIX: physical memory is traded dynamically between VM for application processes and the file system's buffer cache [9]. Since the compression cache must vary in size dynamically as well, Sprite provides a good framework for prototyping the compression cache. The idea of the compression cache should extend naturally to UNIX,[3] Mach, or other systems; in fact, Mach's external pager interface [7] should be an excellent foundation for future work in this area.

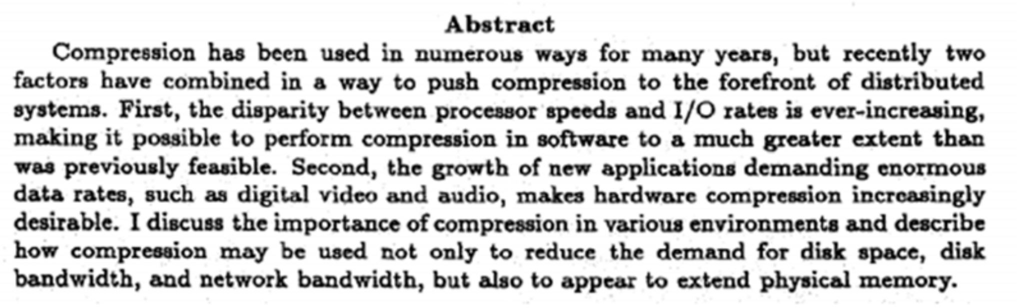
The target environment for this research consists of mobile computers with limited memory and network bandwidth, and with small local disks or no disks at all. Because of the limited availability of Sprite, the compression cache has been prototyped in a workstation environment, running on DECstation 5000/200 workstations, paging to a local RZ57 disk. The Sprite kernel is configurable at boot-time to allow the system to use a variable amount of physical memory, so a 32-Mbyte machine can behave as though it has as little as 12 Mbytes. About 6 Mbytes are used by the kernel for code, page tables, and some forms of tracing that cannot currently be disabled.

Douglis 2, 4

1.4 accessing and decompressing said at least a portion of said boot data in said compressed form from said memory; and	Douglis, as evidenced by the example citations below, discloses "accessing and decompressing said at least a portion of said boot data in said compressed form from said memory; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing and decompressing said at least a portion of said boot data in said compressed form from said memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

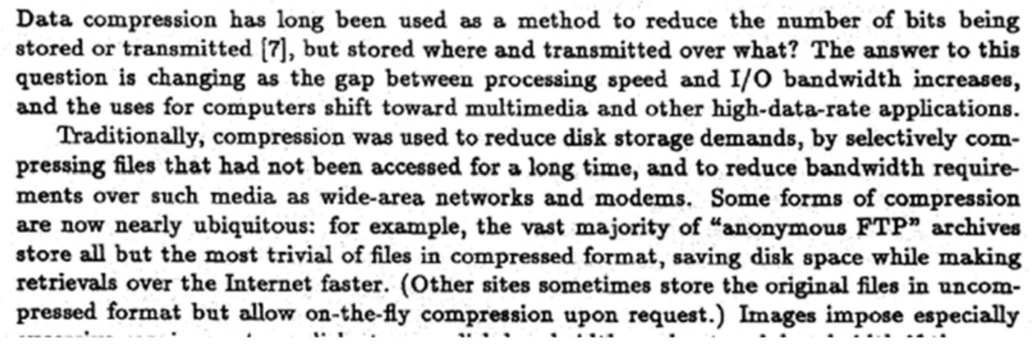
Douglis discloses this limitation:



Abstract

Compression has been used in numerous ways for many years, but recently two factors have combined in a way to push compression to the forefront of distributed systems. First, the disparity between processor speeds and I/O rates is ever-increasing, making it possible to perform compression in software to a much greater extent than was previously feasible. Second, the growth of new applications demanding enormous data rates, such as digital video and audio, makes hardware compression increasingly desirable. I discuss the importance of compression in various environments and describe how compression may be used not only to reduce the demand for disk space, disk bandwidth, and network bandwidth, but also to appear to extend physical memory.

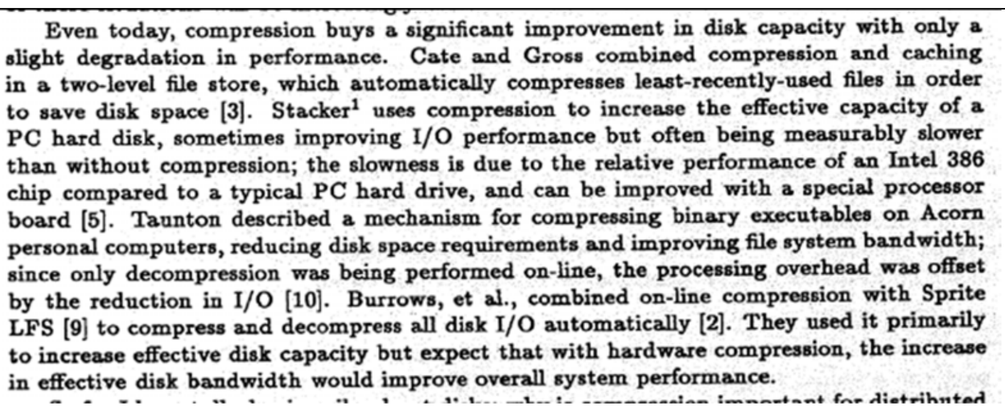
Douglis 1, Abstact



Data compression has long been used as a method to reduce the number of bits being stored or transmitted [7], but stored where and transmitted over what? The answer to this question is changing as the gap between processing speed and I/O bandwidth increases, and the uses for computers shift toward multimedia and other high-data-rate applications.

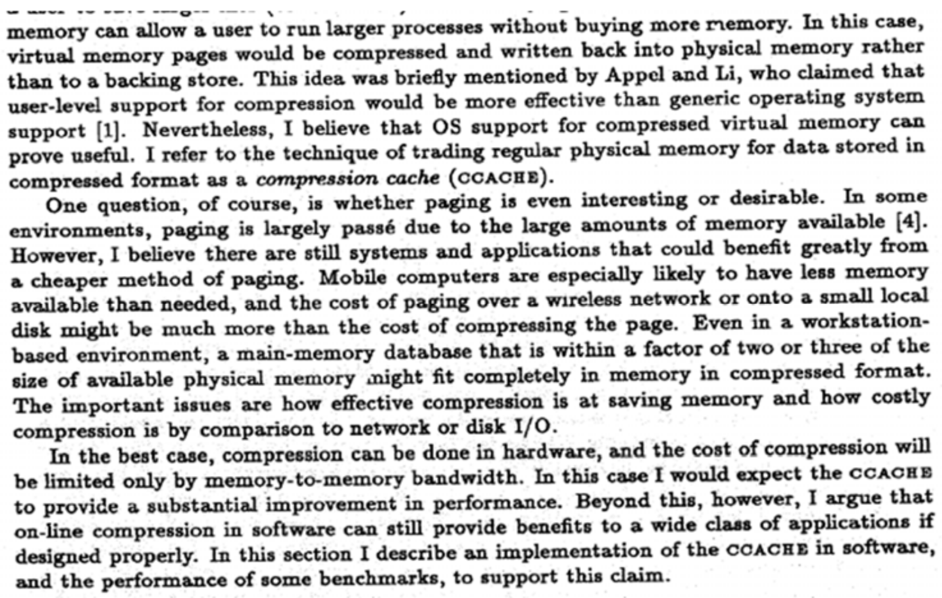
Traditionally, compression was used to reduce disk storage demands, by selectively compressing files that had not been accessed for a long time, and to reduce bandwidth requirements over such media as wide-area networks and modems. Some forms of compression are now nearly ubiquitous: for example, the vast majority of "anonymous FTP" archives store all but the most trivial of files in compressed format, saving disk space while making retrievals over the Internet faster. (Other sites sometimes store the original files in uncompressed format but allow on-the-fly compression upon request.) Images impose especially

Douglis 1, at 1



Even today, compression buys a significant improvement in disk capacity with only a slight degradation in performance. Cate and Gross combined compression and caching in a two-level file store, which automatically compresses least-recently-used files in order to save disk space [3]. Stacker[1] uses compression to increase the effective capacity of a PC hard disk, sometimes improving I/O performance but often being measurably slower than without compression; the slowness is due to the relative performance of an Intel 386 chip compared to a typical PC hard drive, and can be improved with a special processor board [5]. Taunton described a mechanism for compressing binary executables on Acorn personal computers, reducing disk space requirements and improving file system bandwidth; since only decompression was being performed on-line, the processing overhead was offset by the reduction in I/O [10]. Burrows, et al., combined on-line compression with Sprite LFS [9] to compress and decompress all disk I/O automatically [2]. They used it primarily to increase effective disk capacity but expect that with hardware compression, the increase in effective disk bandwidth would improve overall system performance.

Douglis 1, 2

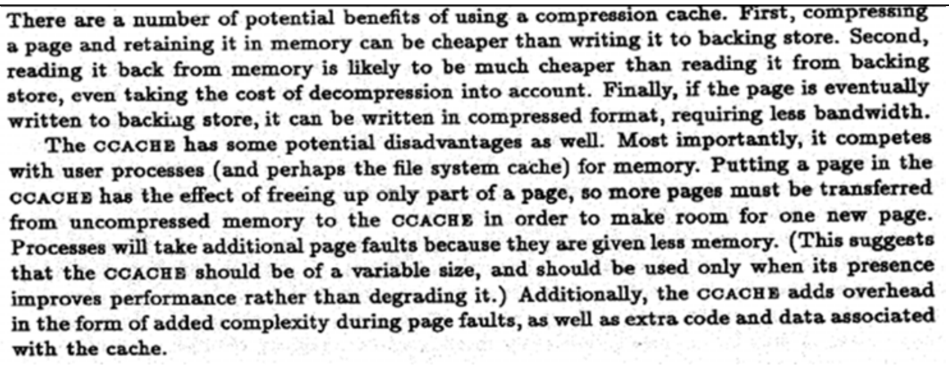


memory can allow a user to run larger processes without buying more memory. In this case, virtual memory pages would be compressed and written back into physical memory rather than to a backing store. This idea was briefly mentioned by Appel and Li, who claimed that user-level support for compression would be more effective than generic operating system support [1]. Nevertheless, I believe that OS support for compressed virtual memory can prove useful. I refer to the technique of trading regular physical memory for data stored in compressed format as a *compression cache* (CCACHE).

One question, of course, is whether paging is even interesting or desirable. In some environments, paging is largely passé due to the large amounts of memory available [4]. However, I believe there are still systems and applications that could benefit greatly from a cheaper method of paging. Mobile computers are especially likely to have less memory available than needed, and the cost of paging over a wireless network or onto a small local disk might be much more than the cost of compressing the page. Even in a workstation-based environment, a main-memory database that is within a factor of two or three of the size of available physical memory might fit completely in memory in compressed format. The important issues are how effective compression is at saving memory and how costly compression is by comparison to network or disk I/O.

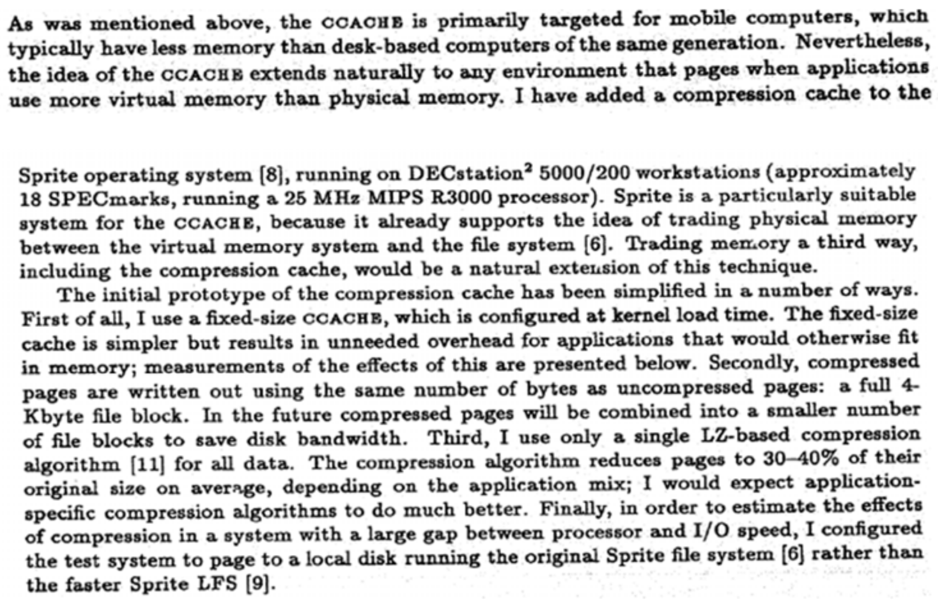
In the best case, compression can be done in hardware, and the cost of compression will be limited only by memory-to-memory bandwidth. In this case I would expect the CCACHE to provide a substantial improvement in performance. Beyond this, however, I argue that on-line compression in software can still provide benefits to a wide class of applications if designed properly. In this section I describe an implementation of the CCACHE in software, and the performance of some benchmarks, to support this claim.

Douglis 1, 2


There are a number of potential benefits of using a compression cache. First, compressing a page and retaining it in memory can be cheaper than writing it to backing store. Second, reading it back from memory is likely to be much cheaper than reading it from backing store, even taking the cost of decompression into account. Finally, if the page is eventually written to backing store, it can be written in compressed format, requiring less bandwidth.

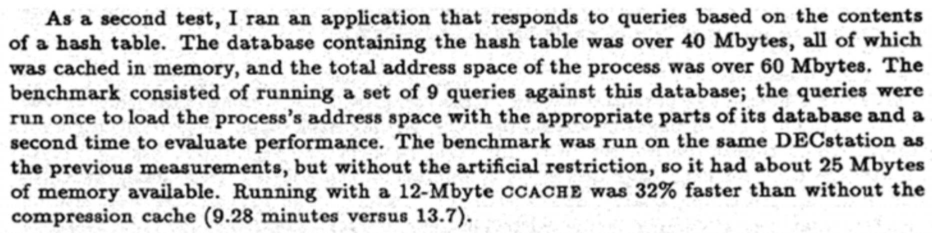
The CCACHE has some potential disadvantages as well. Most importantly, it competes with user processes (and perhaps the file system cache) for memory. Putting a page in the CCACHE has the effect of freeing up only part of a page, so more pages must be transferred from uncompressed memory to the CCACHE in order to make room for one new page. Processes will take additional page faults because they are given less memory. (This suggests that the CCACHE should be of a variable size, and should be used only when its presence improves performance rather than degrading it.) Additionally, the CCACHE adds overhead in the form of added complexity during page faults, as well as extra code and data associated with the cache.

Douglis 1, 3.1

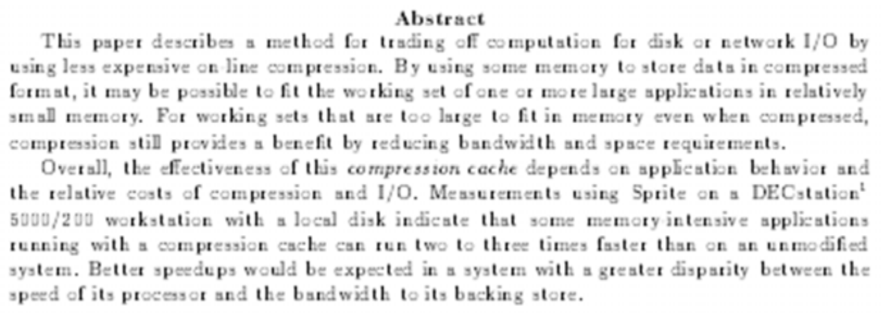

As was mentioned above, the CCACHE is primarily targeted for mobile computers, which typically have less memory than desk-based computers of the same generation. Nevertheless, the idea of the CCACHE extends naturally to any environment that pages when applications use more virtual memory than physical memory. I have added a compression cache to the Sprite operating system [8], running on DECstation[2] 5000/200 workstations (approximately 18 SPECmarks, running a 25 MHz MIPS R3000 processor). Sprite is a particularly suitable system for the CCACHE, because it already supports the idea of trading physical memory between the virtual memory system and the file system [6]. Trading memory a third way, including the compression cache, would be a natural extension of this technique.

The initial prototype of the compression cache has been simplified in a number of ways. First of all, I use a fixed-size CCACHE, which is configured at kernel load time. The fixed-size cache is simpler but results in unneeded overhead for applications that would otherwise fit in memory; measurements of the effects of this are presented below. Secondly, compressed pages are written out using the same number of bytes as uncompressed pages: a full 4-Kbyte file block. In the future compressed pages will be combined into a smaller number of file blocks to save disk bandwidth. Third, I use only a single LZ-based compression algorithm [11] for all data. The compression algorithm reduces pages to 30–40% of their original size on average, depending on the application mix; I would expect application-specific compression algorithms to do much better. Finally, in order to estimate the effects of compression in a system with a large gap between processor and I/O speed, I configured the test system to page to a local disk running the original Sprite file system [6] rather than the faster Sprite LFS [9].

Douglis 1, 3.2


As a second test, I ran an application that responds to queries based on the contents of a hash table. The database containing the hash table was over 40 Mbytes, all of which was cached in memory, and the total address space of the process was over 60 Mbytes. The benchmark consisted of running a set of 9 queries against this database; the queries were run once to load the process's address space with the appropriate parts of its database and a second time to evaluate performance. The benchmark was run on the same DECstation as the previous measurements, but without the artificial restriction, so it had about 25 Mbytes of memory available. Running with a 12-Mbyte CCACHE was 32% faster than without the compression cache (9.28 minutes versus 13.7).

Douglis 1, 3.4

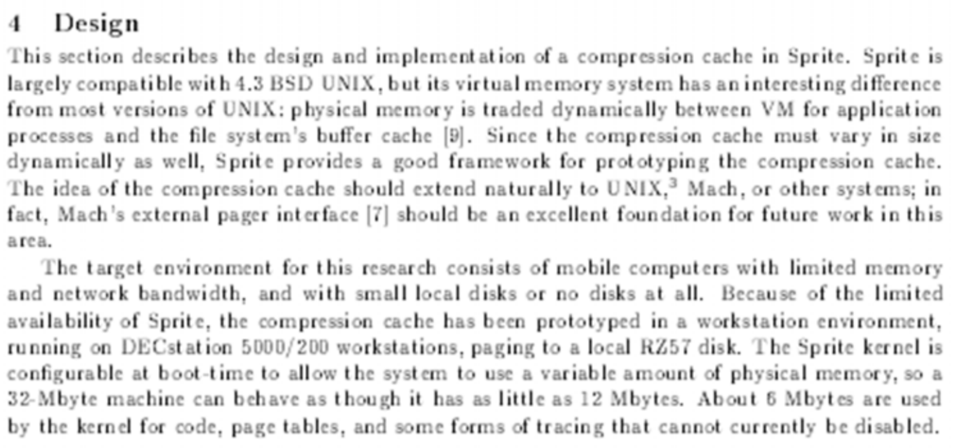


Abstract

This paper describes a method for trading off computation for disk or network I/O by using less expensive on-line compression. By using some memory to store data in compressed format, it may be possible to fit the working set of one or more large applications in relatively small memory. For working sets that are too large to fit in memory even when compressed, compression still provides a benefit by reducing bandwidth and space requirements.

Overall, the effectiveness of this *compression cache* depends on application behavior and the relative costs of compression and I/O. Measurements using Sprite on a DECstation[1] 5000/200 workstation with a local disk indicate that some memory-intensive applications running with a compression cache can run two to three times faster than on an unmodified system. Better speedups would be expected in a system with a greater disparity between the speed of its processor and the bandwidth to its backing store.

Douglis 2, Abstract



4 Design

This section describes the design and implementation of a compression cache in Sprite. Sprite is largely compatible with 4.3 BSD UNIX, but its virtual memory system has an interesting difference from most versions of UNIX: physical memory is traded dynamically between VM for application processes and the file system's buffer cache [9]. Since the compression cache must vary in size dynamically as well, Sprite provides a good framework for prototyping the compression cache. The idea of the compression cache should extend naturally to UNIX,[3] Mach, or other systems; in fact, Mach's external pager interface [7] should be an excellent foundation for future work in this area.

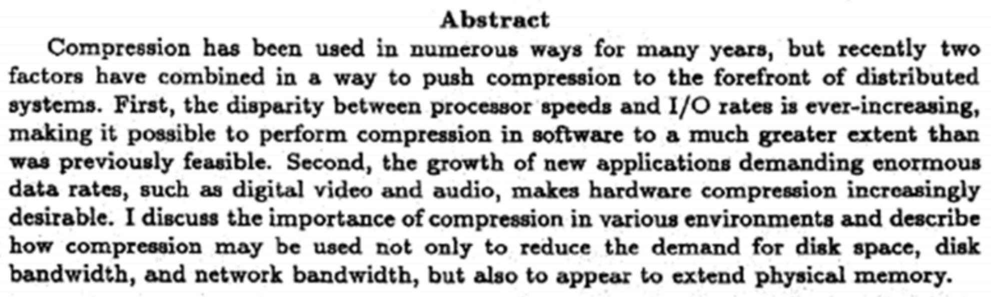
The target environment for this research consists of mobile computers with limited memory and network bandwidth, and with small local disks or no disks at all. Because of the limited availability of Sprite, the compression cache has been prototyped in a workstation environment, running on DECstation 5000/200 workstations, paging to a local RZ57 disk. The Sprite kernel is configurable at boot-time to allow the system to use a variable amount of physical memory, so a 32-Mbyte machine can behave as though it has as little as 12 Mbytes. About 6 Mbytes are used by the kernel for code, page tables, and some forms of tracing that cannot currently be disabled.

Douglis 2, 4

1.5 utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine.	Douglis, as evidenced by the example citations below, discloses "utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

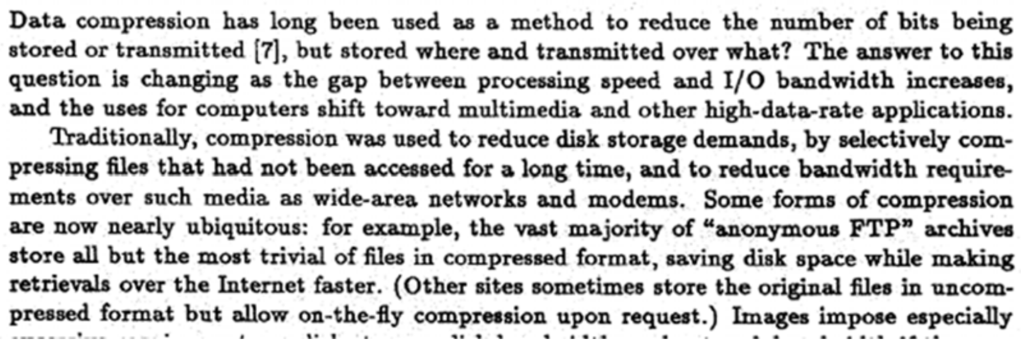
Douglis discloses this limitation:



Abstract

Compression has been used in numerous ways for many years, but recently two factors have combined in a way to push compression to the forefront of distributed systems. First, the disparity between processor speeds and I/O rates is ever-increasing, making it possible to perform compression in software to a much greater extent than was previously feasible. Second, the growth of new applications demanding enormous data rates, such as digital video and audio, makes hardware compression increasingly desirable. I discuss the importance of compression in various environments and describe how compression may be used not only to reduce the demand for disk space, disk bandwidth, and network bandwidth, but also to appear to extend physical memory.

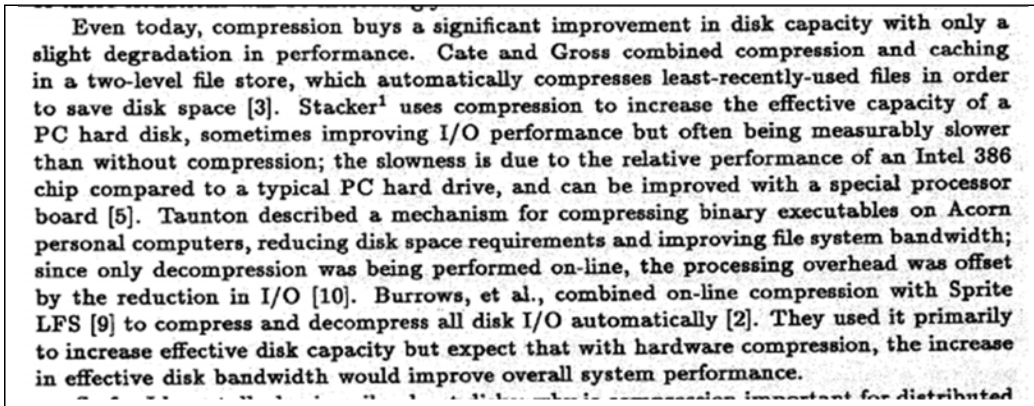
Douglis 1, Abstact



Data compression has long been used as a method to reduce the number of bits being stored or transmitted [7], but stored where and transmitted over what? The answer to this question is changing as the gap between processing speed and I/O bandwidth increases, and the uses for computers shift toward multimedia and other high-data-rate applications.

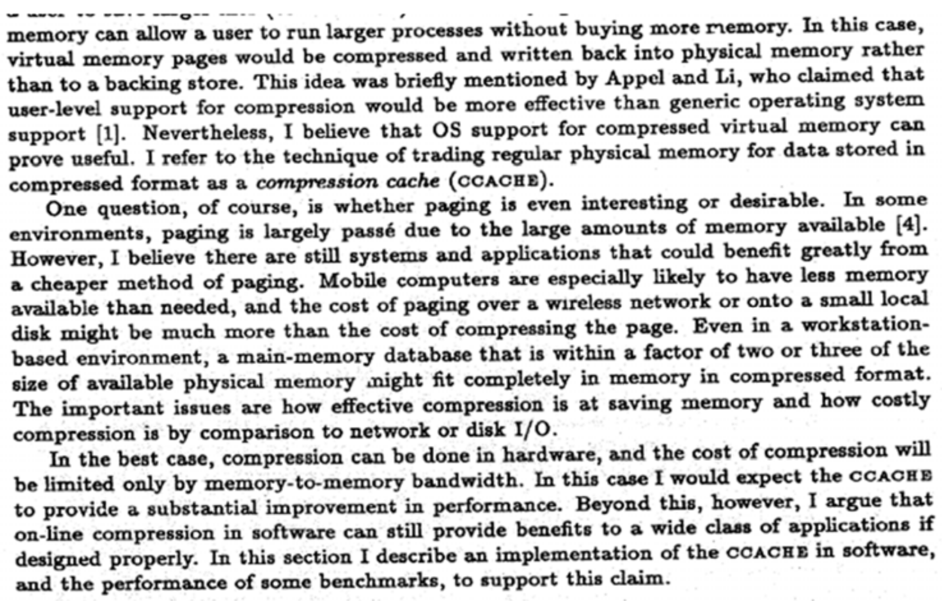
Traditionally, compression was used to reduce disk storage demands, by selectively compressing files that had not been accessed for a long time, and to reduce bandwidth requirements over such media as wide-area networks and modems. Some forms of compression are now nearly ubiquitous: for example, the vast majority of "anonymous FTP" archives store all but the most trivial of files in compressed format, saving disk space while making retrievals over the Internet faster. (Other sites sometimes store the original files in uncompressed format but allow on-the-fly compression upon request.) Images impose especially

Douglis 1, at 1



Even today, compression buys a significant improvement in disk capacity with only a slight degradation in performance. Cate and Gross combined compression and caching in a two-level file store, which automatically compresses least-recently-used files in order to save disk space [3]. Stacker[1] uses compression to increase the effective capacity of a PC hard disk, sometimes improving I/O performance but often being measurably slower than without compression; the slowness is due to the relative performance of an Intel 386 chip compared to a typical PC hard drive, and can be improved with a special processor board [5]. Taunton described a mechanism for compressing binary executables on Acorn personal computers, reducing disk space requirements and improving file system bandwidth; since only decompression was being performed on-line, the processing overhead was offset by the reduction in I/O [10]. Burrows, et al., combined on-line compression with Sprite LFS [9] to compress and decompress all disk I/O automatically [2]. They used it primarily to increase effective disk capacity but expect that with hardware compression, the increase in effective disk bandwidth would improve overall system performance.

Douglis 1, 2

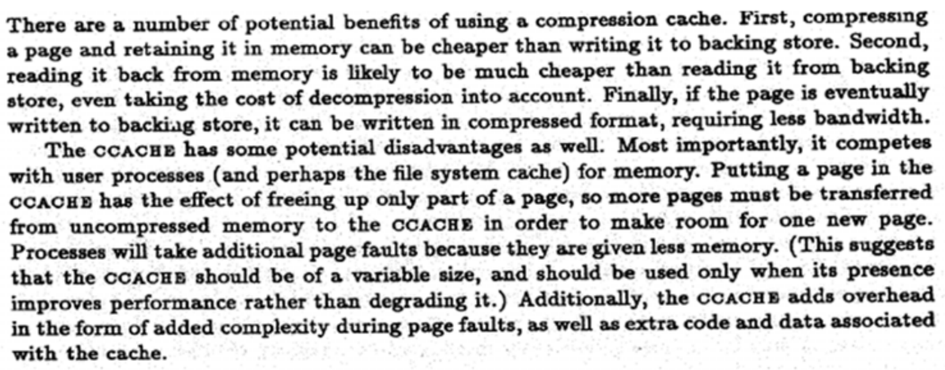


memory can allow a user to run larger processes without buying more memory. In this case, virtual memory pages would be compressed and written back into physical memory rather than to a backing store. This idea was briefly mentioned by Appel and Li, who claimed that user-level support for compression would be more effective than generic operating system support [1]. Nevertheless, I believe that OS support for compressed virtual memory can prove useful. I refer to the technique of trading regular physical memory for data stored in compressed format as a *compression cache* (CCACHE).

One question, of course, is whether paging is even interesting or desirable. In some environments, paging is largely passé due to the large amounts of memory available [4]. However, I believe there are still systems and applications that could benefit greatly from a cheaper method of paging. Mobile computers are especially likely to have less memory available than needed, and the cost of paging over a wireless network or onto a small local disk might be much more than the cost of compressing the page. Even in a workstation-based environment, a main-memory database that is within a factor of two or three of the size of available physical memory might fit completely in memory in compressed format. The important issues are how effective compression is at saving memory and how costly compression is by comparison to network or disk I/O.

In the best case, compression can be done in hardware, and the cost of compression will be limited only by memory-to-memory bandwidth. In this case I would expect the CCACHE to provide a substantial improvement in performance. Beyond this, however, I argue that on-line compression in software can still provide benefits to a wide class of applications if designed properly. In this section I describe an implementation of the CCACHE in software, and the performance of some benchmarks, to support this claim.

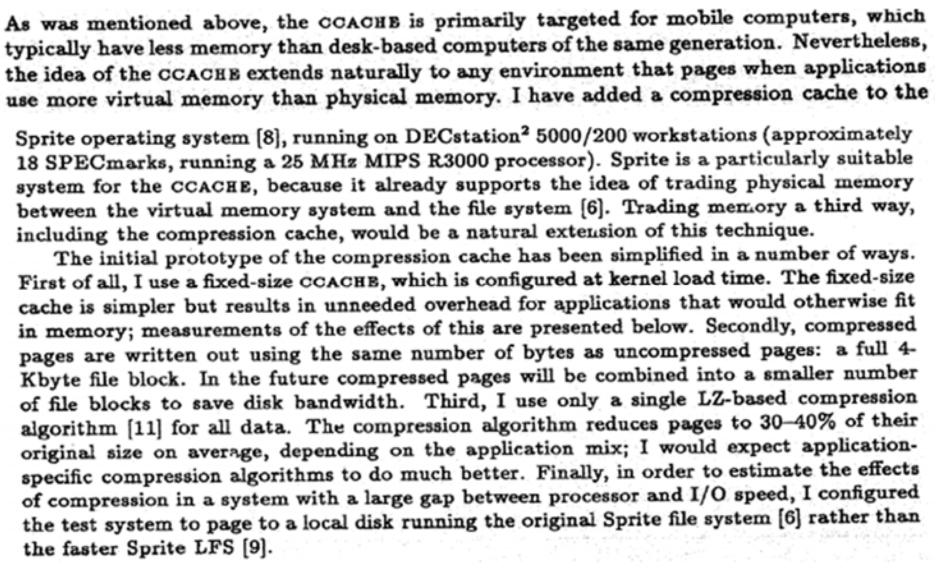
Douglis 1, 2



There are a number of potential benefits of using a compression cache. First, compressing a page and retaining it in memory can be cheaper than writing it to backing store. Second, reading it back from memory is likely to be much cheaper than reading it from backing store, even taking the cost of decompression into account. Finally, if the page is eventually written to backing store, it can be written in compressed format, requiring less bandwidth.

The CCACHE has some potential disadvantages as well. Most importantly, it competes with user processes (and perhaps the file system cache) for memory. Putting a page in the CCACHE has the effect of freeing up only part of a page, so more pages must be transferred from uncompressed memory to the CCACHE in order to make room for one new page. Processes will take additional page faults because they are given less memory. (This suggests that the CCACHE should be of a variable size, and should be used only when its presence improves performance rather than degrading it.) Additionally, the CCACHE adds overhead in the form of added complexity during page faults, as well as extra code and data associated with the cache.

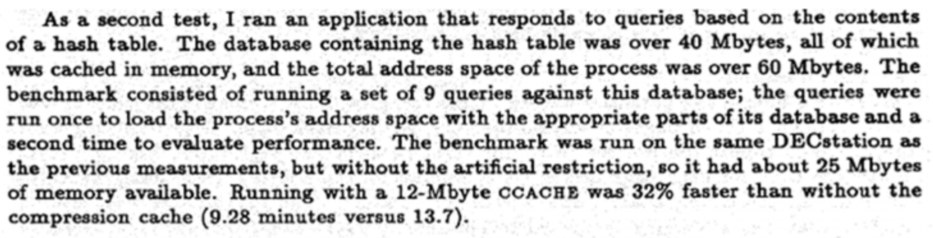
Douglis 1, 3.1



As was mentioned above, the CCACHE is primarily targeted for mobile computers, which typically have less memory than desk-based computers of the same generation. Nevertheless, the idea of the CCACHE extends naturally to any environment that pages when applications use more virtual memory than physical memory. I have added a compression cache to the Sprite operating system [8], running on DECstation[2] 5000/200 workstations (approximately 18 SPECmarks, running a 25 MHz MIPS R3000 processor). Sprite is a particularly suitable system for the CCACHE, because it already supports the idea of trading physical memory between the virtual memory system and the file system [6]. Trading memory a third way, including the compression cache, would be a natural extension of this technique.

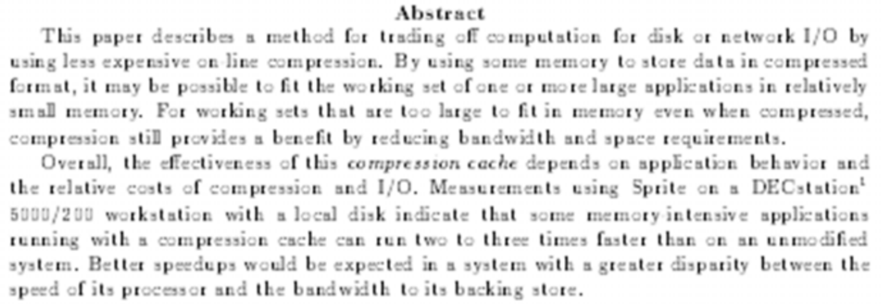
The initial prototype of the compression cache has been simplified in a number of ways. First of all, I use a fixed-size CCACHE, which is configured at kernel load time. The fixed-size cache is simpler but results in unneeded overhead for applications that would otherwise fit in memory; measurements of the effects of this are presented below. Secondly, compressed pages are written out using the same number of bytes as uncompressed pages: a full 4-Kbyte file block. In the future compressed pages will be combined into a smaller number of file blocks to save disk bandwidth. Third, I use only a single LZ-based compression algorithm [11] for all data. The compression algorithm reduces pages to 30–40% of their original size on average, depending on the application mix; I would expect application-specific compression algorithms to do much better. Finally, in order to estimate the effects of compression in a system with a large gap between processor and I/O speed, I configured the test system to page to a local disk running the original Sprite file system [6] rather than the faster Sprite LFS [9].

Douglis 1, 3.2



As a second test, I ran an application that responds to queries based on the contents of a hash table. The database containing the hash table was over 40 Mbytes, all of which was cached in memory, and the total address space of the process was over 60 Mbytes. The benchmark consisted of running a set of 9 queries against this database; the queries were run once to load the process's address space with the appropriate parts of its database and a second time to evaluate performance. The benchmark was run on the same DECstation as the previous measurements, but without the artificial restriction, so it had about 25 Mbytes of memory available. Running with a 12-Mbyte CCACHE was 32% faster than without the compression cache (9.28 minutes versus 13.7).

Douglis 1, 3.4

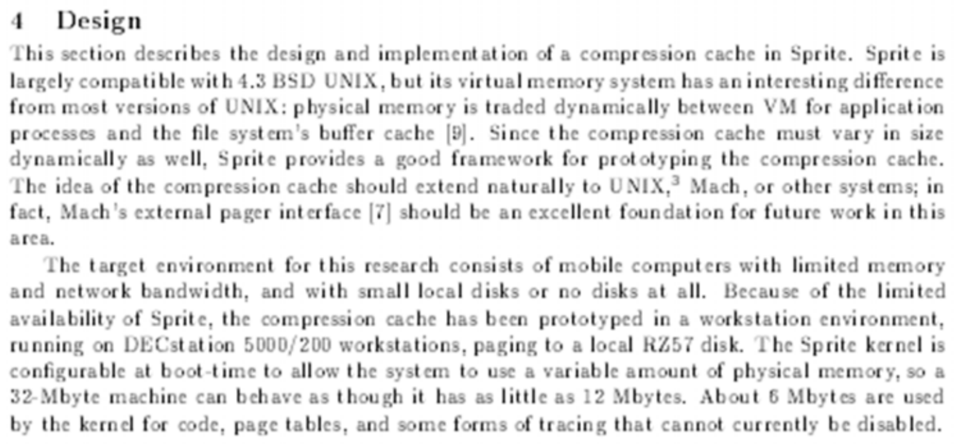


Abstract

This paper describes a method for trading off computation for disk or network I/O by using less expensive on-line compression. By using some memory to store data in compressed format, it may be possible to fit the working set of one or more large applications in relatively small memory. For working sets that are too large to fit in memory even when compressed, compression still provides a benefit by reducing bandwidth and space requirements.

Overall, the effectiveness of this *compression cache* depends on application behavior and the relative costs of compression and I/O. Measurements using Sprite on a DECstation[1] 5000/200 workstation with a local disk indicate that some memory-intensive applications running with a compression cache can run two to three times faster than on an unmodified system. Better speedups would be expected in a system with a greater disparity between the speed of its processor and the bandwidth to its backing store.

Douglis 2, Abstract



4 Design

This section describes the design and implementation of a compression cache in Sprite. Sprite is largely compatible with 4.3 BSD UNIX, but its virtual memory system has an interesting difference from most versions of UNIX: physical memory is traded dynamically between VM for application processes and the file system's buffer cache [9]. Since the compression cache must vary in size dynamically as well, Sprite provides a good framework for prototyping the compression cache. The idea of the compression cache should extend naturally to UNIX,[3] Mach, or other systems; in fact, Mach's external pager interface [7] should be an excellent foundation for future work in this area.

The target environment for this research consists of mobile computers with limited memory and network bandwidth, and with small local disks or no disks at all. Because of the limited availability of Sprite, the compression cache has been prototyped in a workstation environment, running on DECstation 5000/200 workstations, paging to a local RZ57 disk. The Sprite kernel is configurable at boot-time to allow the system to use a variable amount of physical memory, so a 32-Mbyte machine can behave as though it has as little as 12 Mbytes. About 6 Mbytes are used by the kernel for code, page tables, and some forms of tracing that cannot currently be disabled.

Douglis 2, 4

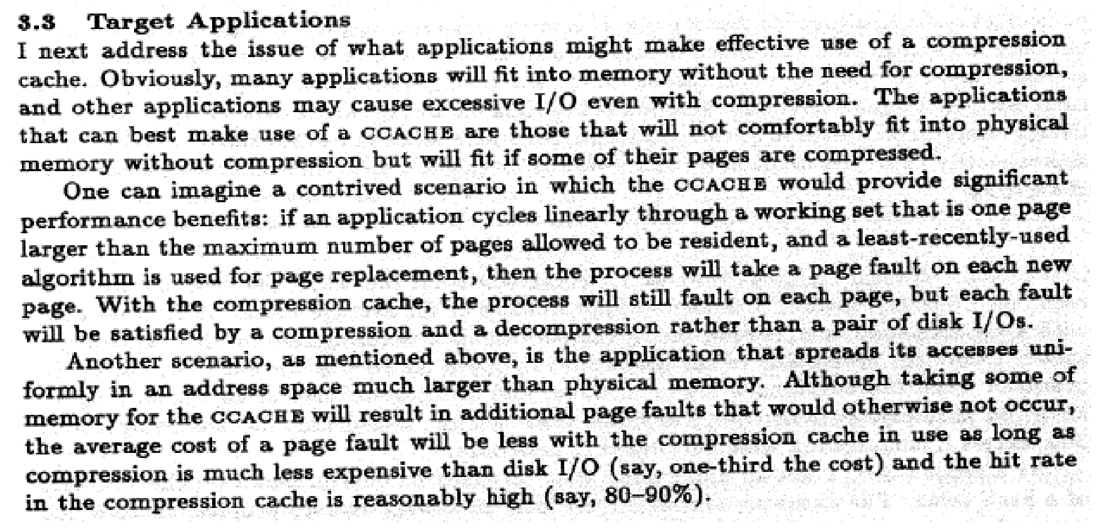
2. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system.	Douglis, as evidenced by the example citations below, discloses “said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Douglis discloses this limitation: *See* Claims 1.1, 1.4, and 1.5 above.	

3. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system.	Douglis, as evidenced by the example citations below, discloses "said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Douglis discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.



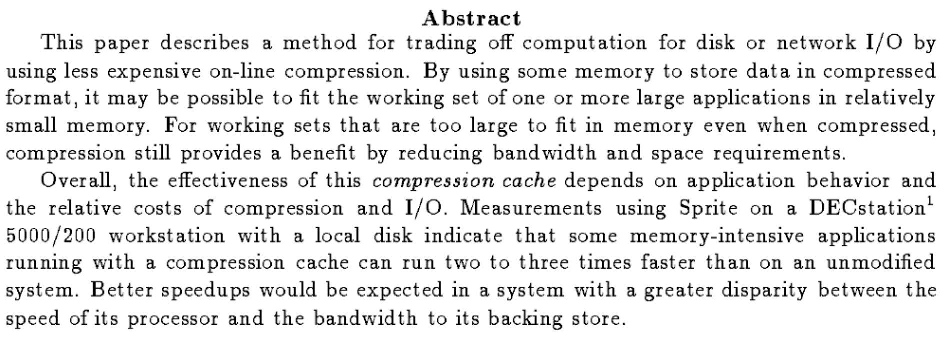
3.3 Target Applications

I next address the issue of what applications might make effective use of a compression cache. Obviously, many applications will fit into memory without the need for compression, and other applications may cause excessive I/O even with compression. The applications that can best make use of a CCACHE are those that will not comfortably fit into physical memory without compression but will fit if some of their pages are compressed.

One can imagine a contrived scenario in which the CCACHE would provide significant performance benefits: if an application cycles linearly through a working set that is one page larger than the maximum number of pages allowed to be resident, and a least-recently-used algorithm is used for page replacement, then the process will take a page fault on each new page. With the compression cache, the process will still fault on each page, but each fault will be satisfied by a compression and a decompression rather than a pair of disk I/Os.

Another scenario, as mentioned above, is the application that spreads its accesses uniformly in an address space much larger than physical memory. Although taking some of memory for the CCACHE will result in additional page faults that would otherwise not occur, the average cost of a page fault will be less with the compression cache in use as long as compression is much less expensive than disk I/O (say, one-third the cost) and the hit rate in the compression cache is reasonably high (say, 80–90%).

Douglis 1



Abstract

This paper describes a method for trading off computation for disk or network I/O by using less expensive on-line compression. By using some memory to store data in compressed format, it may be possible to fit the working set of one or more large applications in relatively small memory. For working sets that are too large to fit in memory even when compressed, compression still provides a benefit by reducing bandwidth and space requirements.

Overall, the effectiveness of this *compression cache* depends on application behavior and the relative costs of compression and I/O. Measurements using Sprite on a DECstation[1] 5000/200 workstation with a local disk indicate that some memory-intensive applications running with a compression cache can run two to three times faster than on an unmodified system. Better speedups would be expected in a system with a greater disparity between the speed of its processor and the bandwidth to its backing store.

Douglis 2

1 Introduction

Over the past decade, the processing power and physical memory size of typical computers have increased dramatically. Even as workstation memory sizes are increasing, however, a new technology trend is pushing toward small memories: mobile computers that are smaller than their desk-top counterparts and are typically configured with significantly less memory. Application designers are sometimes forced to squeeze their applications to fit into available memory, and may not succeed. Therefore, in a general-purpose mobile computer, as with many computers, paging is needed to enable a wider range of applications to run—as long as it can be performed efficiently.

Douglis 2

The potential benefits of the compression cache depend on the relationship between the speed of compression and the I/O bandwidth of the system, as well as the compression ratio (anywhere from barely over 1:1 to about 4:1 in the experiments reported below). If the cost of compressing and copying a page were negligible, and pages compressed well, the compression cache could be used to give a computer the appearance of having additional physical memory. In practice, compressing and copying have costs associated with them, and the benefit of reducing traffic to the backing store is offset by the overhead of the compression cache. Overhead comes not only from the compression itself but from the additional page faults an application will experience when some memory is used for compressed pages (as well as the data structures used to support compressed pages). Furthermore, as mentioned above, not all applications compress well: for poorly suited applications, the effort to compress memory will be wasted and degrade rather than improve performance. Thus, depending on the application and the hardware environment, the benefits of reduced I/O may outweigh the costs of compression and additional faults, or vice-versa. Configuring the compression cache to improve performance in the first case while staying out of the way in the second case is an interesting, and difficult, problem.

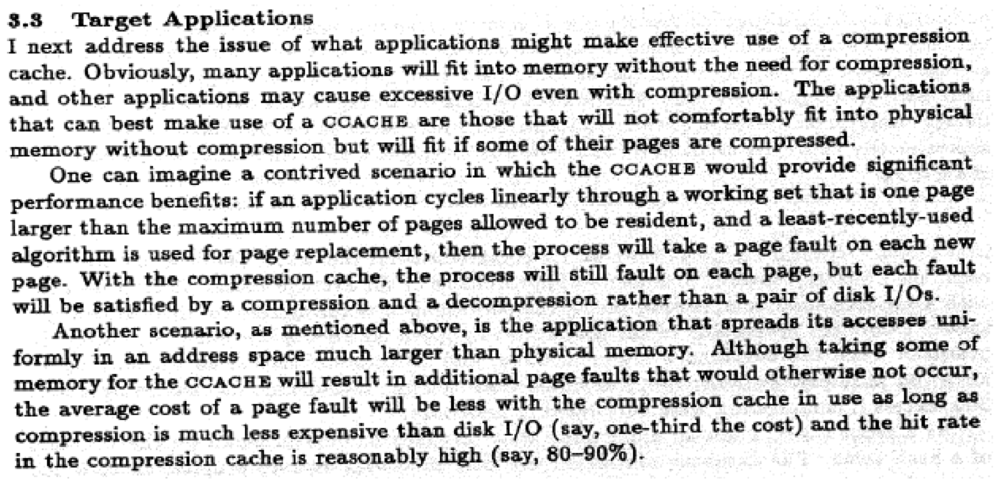
Douglis 2

4. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an application program and an operating system of said computer system.	Douglis, as evidenced by the example citations below, discloses "said decompressed at least a portion of said boot data comprises program code associated with an application program and an operating system of said computer system."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Douglis discloses this limitation:

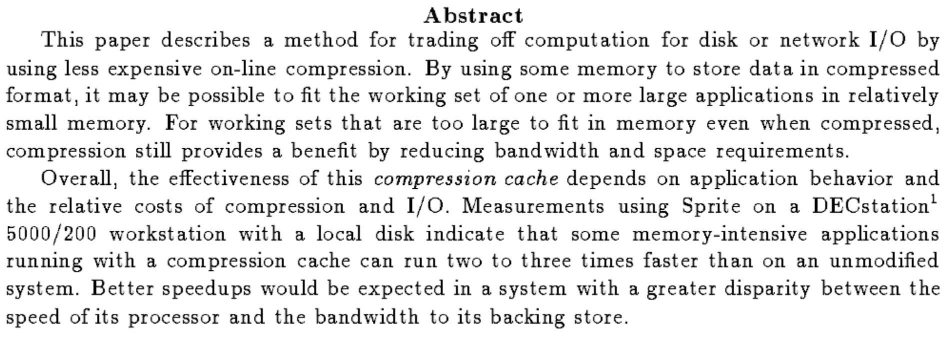
See Claims 1.1, 1.3, and 1.4 above.


3.3 Target Applications
I next address the issue of what applications might make effective use of a compression cache. Obviously, many applications will fit into memory without the need for compression, and other applications may cause excessive I/O even with compression. The applications that can best make use of a CCACHE are those that will not comfortably fit into physical memory without compression but will fit if some of their pages are compressed.

One can imagine a contrived scenario in which the CCACHE would provide significant performance benefits: if an application cycles linearly through a working set that is one page larger than the maximum number of pages allowed to be resident, and a least-recently-used algorithm is used for page replacement, then the process will take a page fault on each new page. With the compression cache, the process will still fault on each page, but each fault will be satisfied by a compression and a decompression rather than a pair of disk I/Os.

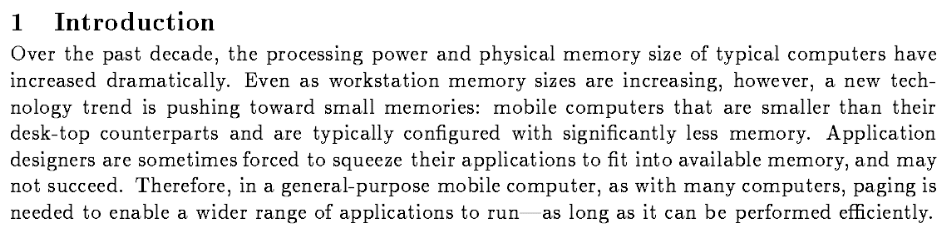
Another scenario, as mentioned above, is the application that spreads its accesses uniformly in an address space much larger than physical memory. Although taking some of memory for the CCACHE will result in additional page faults that would otherwise not occur, the average cost of a page fault will be less with the compression cache in use as long as compression is much less expensive than disk I/O (say, one-third the cost) and the hit rate in the compression cache is reasonably high (say, 80–90%).

Douglis 1


Abstract
This paper describes a method for trading off computation for disk or network I/O by using less expensive on-line compression. By using some memory to store data in compressed format, it may be possible to fit the working set of one or more large applications in relatively small memory. For working sets that are too large to fit in memory even when compressed, compression still provides a benefit by reducing bandwidth and space requirements.

Overall, the effectiveness of this *compression cache* depends on application behavior and the relative costs of compression and I/O. Measurements using Sprite on a DECstation[1] 5000/200 workstation with a local disk indicate that some memory-intensive applications running with a compression cache can run two to three times faster than on an unmodified system. Better speedups would be expected in a system with a greater disparity between the speed of its processor and the bandwidth to its backing store.

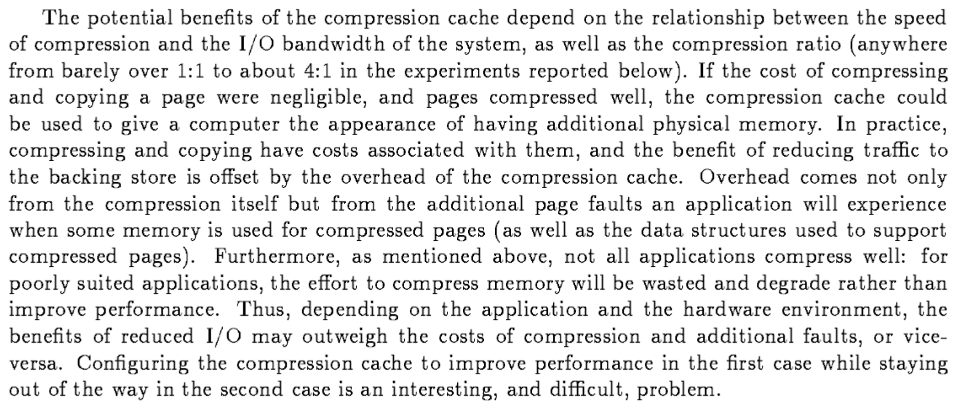
Douglis 2



1 Introduction

Over the past decade, the processing power and physical memory size of typical computers have increased dramatically. Even as workstation memory sizes are increasing, however, a new technology trend is pushing toward small memories: mobile computers that are smaller than their desk-top counterparts and are typically configured with significantly less memory. Application designers are sometimes forced to squeeze their applications to fit into available memory, and may not succeed. Therefore, in a general-purpose mobile computer, as with many computers, paging is needed to enable a wider range of applications to run—as long as it can be performed efficiently.

Douglis 2



The potential benefits of the compression cache depend on the relationship between the speed of compression and the I/O bandwidth of the system, as well as the compression ratio (anywhere from barely over 1:1 to about 4:1 in the experiments reported below). If the cost of compressing and copying a page were negligible, and pages compressed well, the compression cache could be used to give a computer the appearance of having additional physical memory. In practice, compressing and copying have costs associated with them, and the benefit of reducing traffic to the backing store is offset by the overhead of the compression cache. Overhead comes not only from the compression itself but from the additional page faults an application will experience when some memory is used for compressed pages (as well as the data structures used to support compressed pages). Furthermore, as mentioned above, not all applications compress well: for poorly suited applications, the effort to compress memory will be wasted and degrade rather than improve performance. Thus, depending on the application and the hardware environment, the benefits of reduced I/O may outweigh the costs of compression and additional faults, or vice-versa. Configuring the compression cache to improve performance in the first case while staying out of the way in the second case is an interesting, and difficult, problem.

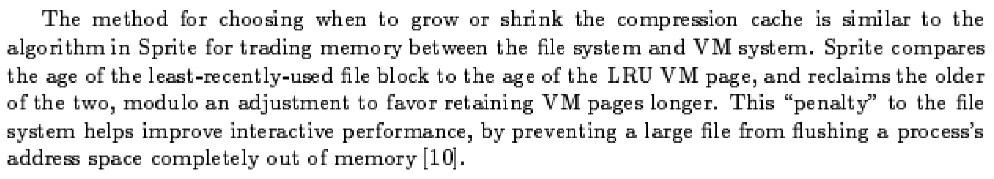
Douglis 2

5. The method of claim 1, wherein said preloading is performed by a data storage controller connected to said boot device.	Douglis, as evidenced by the example citations below, discloses "said preloading is performed by a data storage controller connected to said boot device."

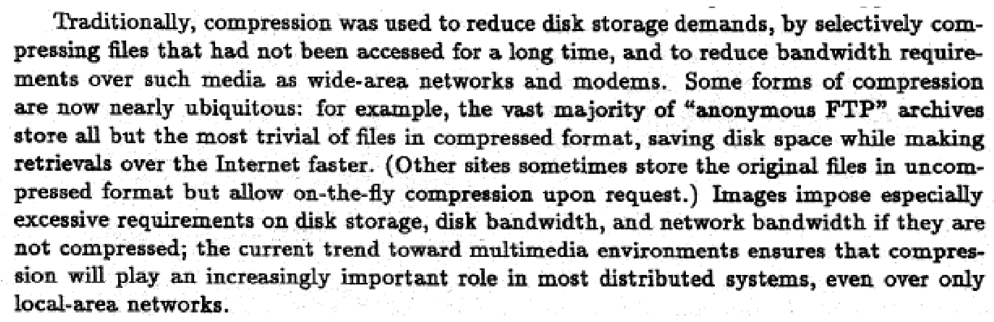
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said preloading is performed by a data storage controller connected to said boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Douglis discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.


The method for choosing when to grow or shrink the compression cache is similar to the algorithm in Sprite for trading memory between the file system and VM system. Sprite compares the age of the least-recently-used file block to the age of the LRU VM page, and reclaims the older of the two, modulo an adjustment to favor retaining VM pages longer. This "penalty" to the file system helps improve interactive performance, by preventing a large file from flushing a process's address space completely out of memory [10].

Douglis 2


Traditionally, compression was used to reduce disk storage demands, by selectively compressing files that had not been accessed for a long time, and to reduce bandwidth requirements over such media as wide-area networks and modems. Some forms of compression are now nearly ubiquitous: for example, the vast majority of "anonymous FTP" archives store all but the most trivial of files in compressed format, saving disk space while making retrievals over the Internet faster. (Other sites sometimes store the original files in uncompressed format but allow on-the-fly compression upon request.) Images impose especially excessive requirements on disk storage, disk bandwidth, and network bandwidth if they are not compressed; the current trend toward multimedia environments ensures that compression will play an increasingly important role in most distributed systems, even over only local-area networks.

Douglis 1

6. The method of claim 1, further comprising updating the list of boot data.	Douglis, as evidenced by the example citations below, discloses "updating the list of boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

Appendix B30
Invalidity of U.S. Patent 8,090,936 based on Douglis

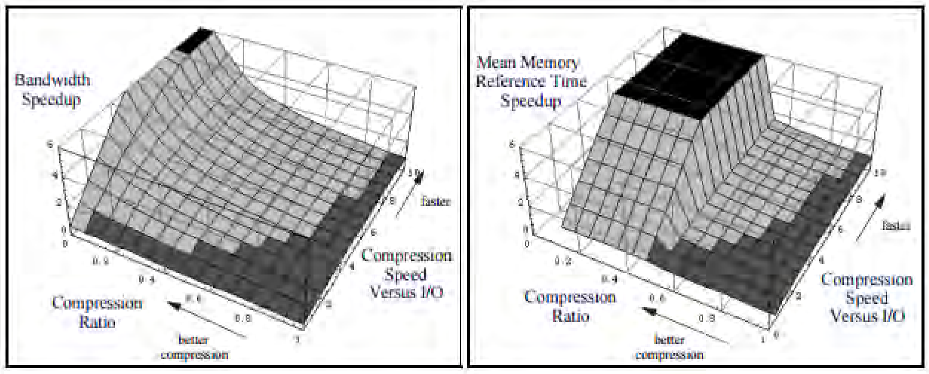
8. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form.	Douglis, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Douglis discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

The Compression Cache ...

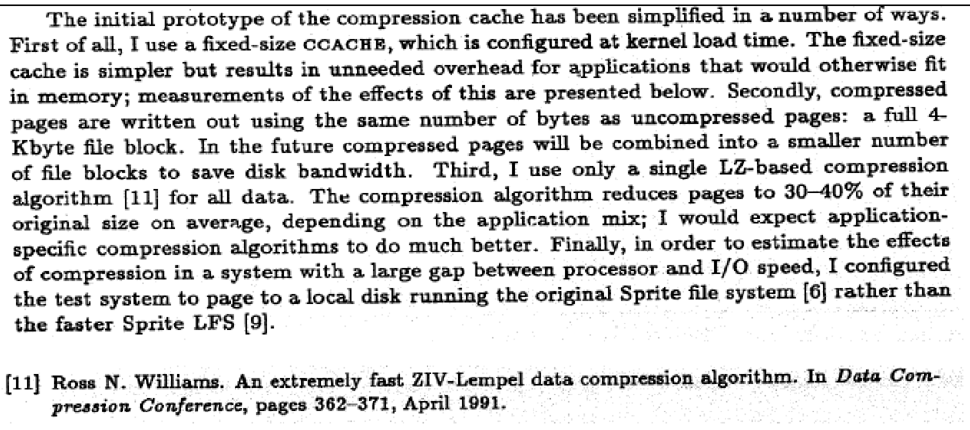


(a) Transferring compressed pages to backing store. (b) Keeping compressed pages in memory.

Figure 1: Performance of compressing pages, modeled analytically. Speedups are shown as a function of the compression ratio (fraction of bytes left after compression) and the speed of compression relative to I/O. Decompression is assumed to be twice as fast as compression, as is roughly the case for algorithms such as LZRW1 [16]. There are three regions of speedup: the dark black areas at the top left show speedups that go off the top of the scale (6-fold improvement); the light areas show speedups of 1–6 relative to no compression, and the darker areas to the right show data points at which a slowdown would result.

[16] Ross N. Williams. An extremely fast ZIV-Lempel data compression algorithm. In *Data Compression Conference*, pages 362–371, April 1991.

Douglis 2


The initial prototype of the compression cache has been simplified in a number of ways. First of all, I use a fixed-size CCACHE, which is configured at kernel load time. The fixed-size cache is simpler but results in unneeded overhead for applications that would otherwise fit in memory; measurements of the effects of this are presented below. Secondly, compressed pages are written out using the same number of bytes as uncompressed pages: a full 4-Kbyte file block. In the future compressed pages will be combined into a smaller number of file blocks to save disk bandwidth. Third, I use only a single LZ-based compression algorithm [11] for all data. The compression algorithm reduces pages to 30–40% of their original size on average, depending on the application mix; I would expect application-specific compression algorithms to do much better. Finally, in order to estimate the effects of compression in a system with a large gap between processor and I/O speed, I configured the test system to page to a local disk running the original Sprite file system [6] rather than the faster Sprite LFS [9].

[11] Ross N. Williams. An extremely fast ZIV-Lempel data compression algorithm. In *Data Compression Conference*, pages 362–371, April 1991.

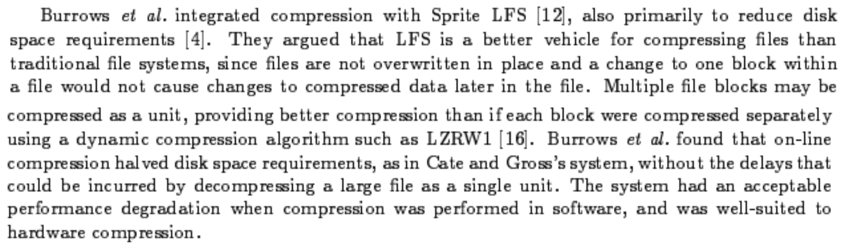
Douglis 1

9. The method of claim 1, wherein a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form.	Douglis, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Douglis discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.


Burrows *et al.* integrated compression with Sprite LFS [12], also primarily to reduce disk space requirements [4]. They argued that LFS is a better vehicle for compressing files than traditional file systems, since files are not overwritten in place and a change to one block within a file would not cause changes to compressed data later in the file. Multiple file blocks may be compressed as a unit, providing better compression than if each block were compressed separately using a dynamic compression algorithm such as LZRW1 [16]. Burrows *et al.* found that on-line compression halved disk space requirements, as in Cate and Gross's system, without the delays that could be incurred by decompressing a large file as a single unit. The system had an acceptable performance degradation when compression was performed in software, and was well-suited to hardware compression.

Douglis 2

11.1. a processor;	Douglis, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claim 1.2 above.	

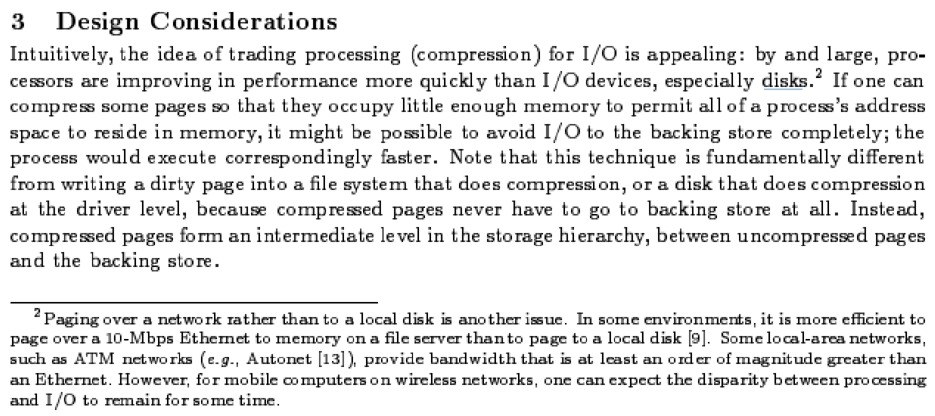
11.2. a memory; and	Douglis, as evidenced by the example citations below, discloses "a memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claims 1.3, and 1.4 above.	

11.3.1 a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device,	Douglis, as evidenced by the example citations below, discloses "a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Douglis discloses this limitation:

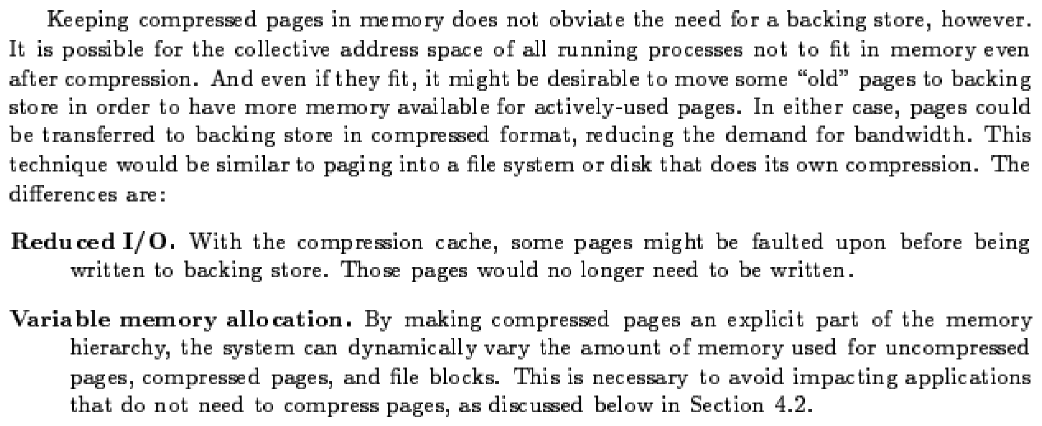
See Claims 1.1, 1.3, 1.4 above.



3 Design Considerations

Intuitively, the idea of trading processing (compression) for I/O is appealing: by and large, processors are improving in performance more quickly than I/O devices, especially disks.[2] If one can compress some pages so that they occupy little enough memory to permit all of a process's address space to reside in memory, it might be possible to avoid I/O to the backing store completely; the process would execute correspondingly faster. Note that this technique is fundamentally different from writing a dirty page into a file system that does compression, or a disk that does compression at the driver level, because compressed pages never have to go to backing store at all. Instead, compressed pages form an intermediate level in the storage hierarchy, between uncompressed pages and the backing store.

[2] Paging over a network rather than to a local disk is another issue. In some environments, it is more efficient to page over a 10-Mbps Ethernet to memory on a file server than to page to a local disk [9]. Some local-area networks, such as ATM networks (*e.g.*, Autonet [13]), provide bandwidth that is at least an order of magnitude greater than an Ethernet. However, for mobile computers on wireless networks, one can expect the disparity between processing and I/O to remain for some time.


Keeping compressed pages in memory does not obviate the need for a backing store, however. It is possible for the collective address space of all running processes not to fit in memory even after compression. And even if they fit, it might be desirable to move some "old" pages to backing store in order to have more memory available for actively-used pages. In either case, pages could be transferred to backing store in compressed format, reducing the demand for bandwidth. This technique would be similar to paging into a file system or disk that does its own compression. The differences are:

Reduced I/O. With the compression cache, some pages might be faulted upon before being written to backing store. Those pages would no longer need to be written.

Variable memory allocation. By making compressed pages an explicit part of the memory hierarchy, the system can dynamically vary the amount of memory used for uncompressed pages, compressed pages, and file blocks. This is necessary to avoid impacting applications that do not need to compress pages, as discussed below in Section 4.2.

Douglis 2

11.3.2 said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system; and,	Douglis, as evidenced by the example citations below, discloses "said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claims 1.3, 1.4, and 1.5 above.	

11.4 a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data.	Douglis, as evidenced by the example citations below, discloses "a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claims 1.1 and 1.5 above.	

12. The system of claim 11, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program.	Douglis, as evidenced by the example citations below, discloses "said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claims 1.1, 3, and 11.3.1 above.	

Appendix B30
Invalidity of U.S. Patent 8,090,936 based on Douglis

13. The system of claim 11, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system.	Douglis, as evidenced by the example citations below, discloses "wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claims 1.1, 1.3, 3, 5 and 11.3.1 and 11.3.2 above.	

15. The system of claim 11, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form.	Douglis, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claims 1, 8, and 11 above.	

16. The system of claim 11, wherein a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form.	Douglis, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Douglis discloses this limitation: *See* Claims 1, 9, and 11 above.	

Appendix B31
Invalidity of U.S. Patent 8,090,936 based on Grove

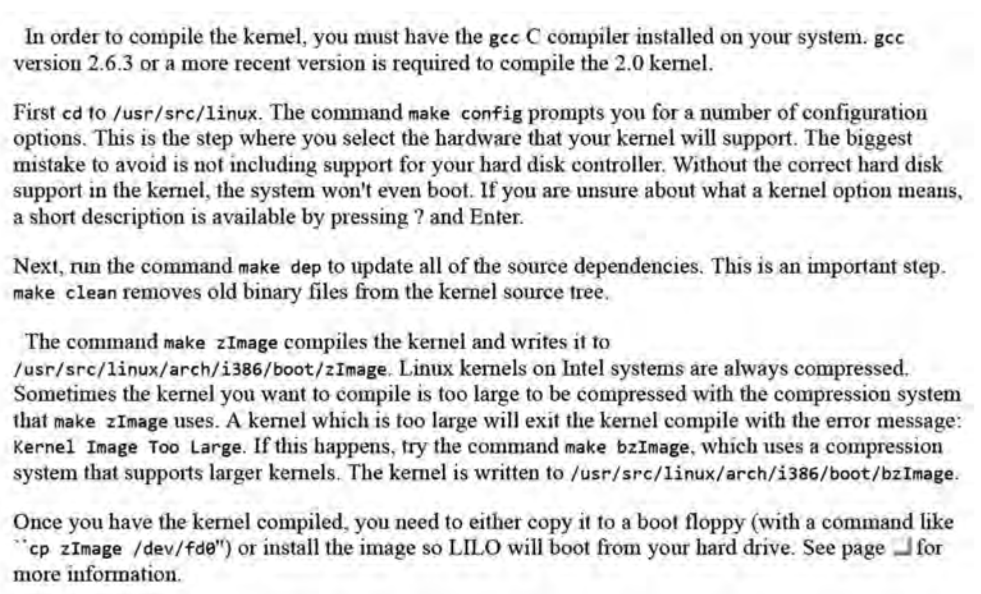
The publication Grove "System Administration," LINUX, Mar 1998. ("Grove") invalidates claims 1-6, 8-9, 11-13, and 15-16 of United States Patent No. 8,090,936 ("the '936 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '936 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

1.1 maintaining a list of boot data used for booting a computer system, wherein at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device;	Grove, as evidenced by the exemplary citations below, discloses "maintaining a list of boot data used for booting a computer system, wherein at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system, wherein at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Grove discloses this claim limitation:


In order to compile the kernel, you must have the `gcc` C compiler installed on your system. `gcc` version 2.6.3 or a more recent version is required to compile the 2.0 kernel.

First `cd` to `/usr/src/linux`. The command `make config` prompts you for a number of configuration options. This is the step where you select the hardware that your kernel will support. The biggest mistake to avoid is not including support for your hard disk controller. Without the correct hard disk support in the kernel, the system won't even boot. If you are unsure about what a kernel option means, a short description is available by pressing ? and Enter.

Next, run the command `make dep` to update all of the source dependencies. This is an important step. `make clean` removes old binary files from the kernel source tree.

The command `make zImage` compiles the kernel and writes it to `/usr/src/linux/arch/i386/boot/zImage`. Linux kernels on Intel systems are always compressed. Sometimes the kernel you want to compile is too large to be compressed with the compression system that `make zImage` uses. A kernel which is too large will exit the kernel compile with the error message: `Kernel Image Too Large`. If this happens, try the command `make bzImage`, which uses a compression system that supports larger kernels. The kernel is written to `/usr/src/linux/arch/i386/boot/bzImage`.

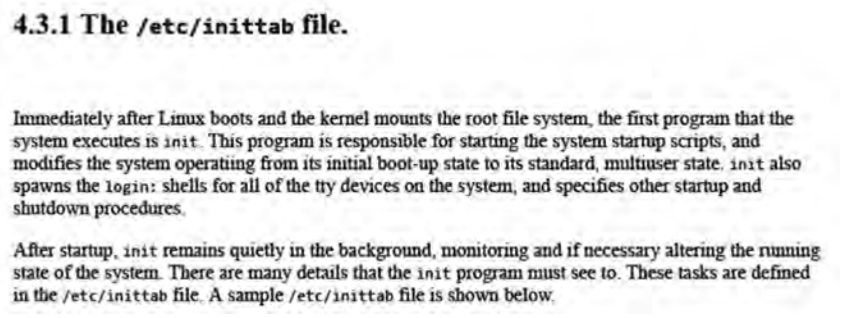
Once you have the kernel compiled, you need to either copy it to a boot floppy (with a command like ``cp zImage /dev/fd0`') or install the image so LILO will boot from your hard drive. See page ☐ for more information.

Grove, 4.9.1

1.2 initializing a central processing unit of said computer system;	Grove, as evidenced by the example citations below, discloses "initializing a central processing unit of said computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Grove discloses this claim limitation:

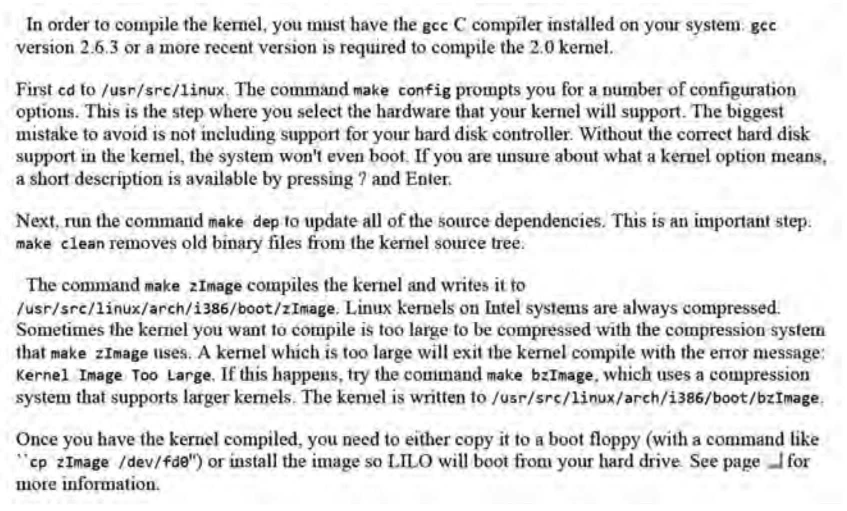


4.3.1 The `/etc/inittab` file.

Immediately after Linux boots and the kernel mounts the root file system, the first program that the system executes is `init`. This program is responsible for starting the system startup scripts, and modifies the system operatiing from its initial boot-up state to its standard, multiuser state. `init` also spawns the `login:` shells for all of the tty devices on the system, and specifies other startup and shutdown procedures.

After startup, `init` remains quietly in the background, monitoring and if necessary altering the running state of the system. There are many details that the `init` program must see to. These tasks are defined in the `/etc/inittab` file. A sample `/etc/inittab` file is shown below.

Grove, 4.3.1



In order to compile the kernel, you must have the `gcc` C compiler installed on your system. `gcc` version 2.6.3 or a more recent version is required to compile the 2.0 kernel.

First `cd` to `/usr/src/linux`. The command `make config` prompts you for a number of configuration options. This is the step where you select the hardware that your kernel will support. The biggest mistake to avoid is not including support for your hard disk controller. Without the correct hard disk support in the kernel, the system won't even boot. If you are unsure about what a kernel option means, a short description is available by pressing ? and Enter.

Next, run the command `make dep` to update all of the source dependencies. This is an important step. `make clean` removes old binary files from the kernel source tree.

The command `make zImage` compiles the kernel and writes it to `/usr/src/linux/arch/i386/boot/zImage`. Linux kernels on Intel systems are always compressed. Sometimes the kernel you want to compile is too large to be compressed with the compression system that `make zImage` uses. A kernel which is too large will exit the kernel compile with the error message: `Kernel Image Too Large`. If this happens, try the command `make bzImage`, which uses a compression system that supports larger kernels. The kernel is written to `/usr/src/linux/arch/i386/boot/bzImage`.

Once you have the kernel compiled, you need to either copy it to a boot floppy (with a command like ``cp zImage /dev/fd0'') or install the image so LILO will boot from your hard drive. See page _| for more information.

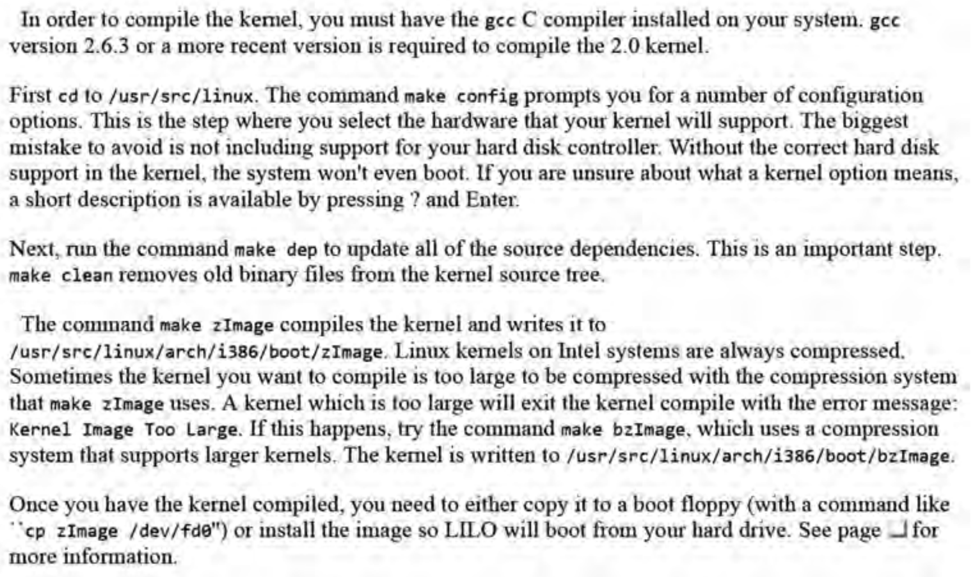
Grove, 4.9.1

Appendix B31
Invalidity of U.S. Patent 8,090,936 based on Grove

1.3 preloading said at least a portion of said boot data in compressed form from said boot device to a memory;	Grove, as evidenced by the example citations below, discloses "preloading said at least a portion of said boot data in compressed form from said boot device to a memory;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading said at least a portion of said boot data in compressed form from said boot device to a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Grove discloses this claim limitation:



In order to compile the kernel, you must have the gcc C compiler installed on your system. gcc version 2.6.3 or a more recent version is required to compile the 2.0 kernel.

First cd to /usr/src/linux. The command make config prompts you for a number of configuration options. This is the step where you select the hardware that your kernel will support. The biggest mistake to avoid is not including support for your hard disk controller. Without the correct hard disk support in the kernel, the system won't even boot. If you are unsure about what a kernel option means, a short description is available by pressing ? and Enter.

Next, run the command make dep to update all of the source dependencies. This is an important step. make clean removes old binary files from the kernel source tree.

The command make zImage compiles the kernel and writes it to /usr/src/linux/arch/i386/boot/zImage. Linux kernels on Intel systems are always compressed. Sometimes the kernel you want to compile is too large to be compressed with the compression system that make zImage uses. A kernel which is too large will exit the kernel compile with the error message: Kernel Image Too Large. If this happens, try the command make bzImage, which uses a compression system that supports larger kernels. The kernel is written to /usr/src/linux/arch/i386/boot/bzImage.

Once you have the kernel compiled, you need to either copy it to a boot floppy (with a command like ``cp zImage /dev/fd0'') or install the image so LILO will boot from your hard drive. See page ␣ for more information.

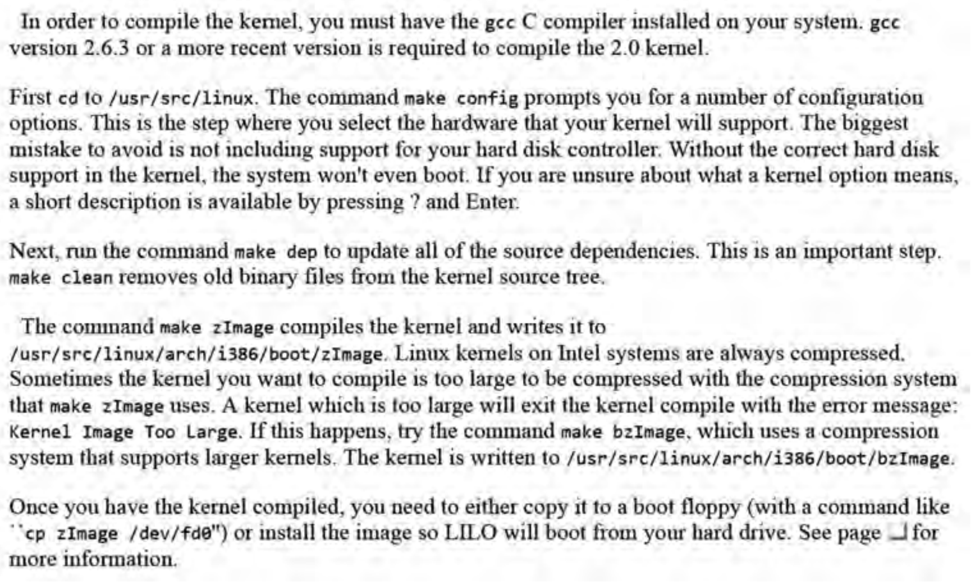
Grove, 4.9.1

Appendix B31
Invalidity of U.S. Patent 8,090,936 based on Grove

1.4 accessing and decompressing said at least a portion of said boot data in said compressed form from said memory; and	Grove, as evidenced by the example citations below, discloses "accessing and decompressing said at least a portion of said boot data in said compressed form from said memory; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing and decompressing said at least a portion of said boot data in said compressed form from said memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Grove discloses this claim limitation:



In order to compile the kernel, you must have the `gcc` C compiler installed on your system. `gcc` version 2.6.3 or a more recent version is required to compile the 2.0 kernel.

First `cd` to `/usr/src/linux`. The command `make config` prompts you for a number of configuration options. This is the step where you select the hardware that your kernel will support. The biggest mistake to avoid is not including support for your hard disk controller. Without the correct hard disk support in the kernel, the system won't even boot. If you are unsure about what a kernel option means, a short description is available by pressing ? and Enter.

Next, run the command `make dep` to update all of the source dependencies. This is an important step. `make clean` removes old binary files from the kernel source tree.

The command `make zImage` compiles the kernel and writes it to `/usr/src/linux/arch/i386/boot/zImage`. Linux kernels on Intel systems are always compressed. Sometimes the kernel you want to compile is too large to be compressed with the compression system that `make zImage` uses. A kernel which is too large will exit the kernel compile with the error message: `Kernel Image Too Large`. If this happens, try the command `make bzImage`, which uses a compression system that supports larger kernels. The kernel is written to `/usr/src/linux/arch/i386/boot/bzImage`.

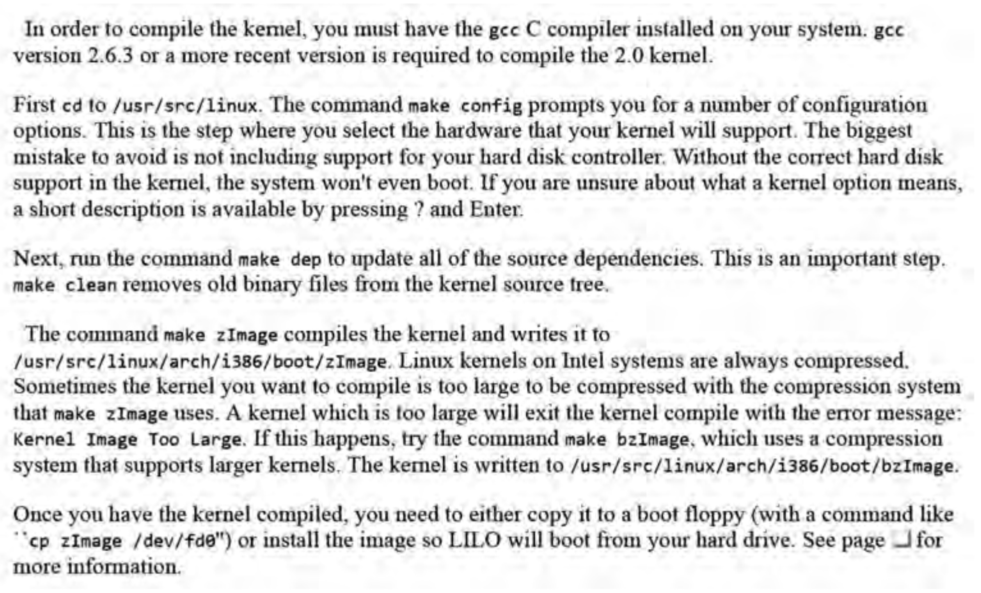
Once you have the kernel compiled, you need to either copy it to a boot floppy (with a command like ``cp zImage /dev/fd0") or install the image so LILO will boot from your hard drive. See page ␣ for more information.

Grove, 4.9.1

1.5 utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine.	Grove, as evidenced by the example citations below, discloses "utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Grove discloses this claim limitation:



In order to compile the kernel, you must have the `gcc` C compiler installed on your system. `gcc` version 2.6.3 or a more recent version is required to compile the 2.0 kernel.

First `cd` to `/usr/src/linux`. The command `make config` prompts you for a number of configuration options. This is the step where you select the hardware that your kernel will support. The biggest mistake to avoid is not including support for your hard disk controller. Without the correct hard disk support in the kernel, the system won't even boot. If you are unsure about what a kernel option means, a short description is available by pressing ? and Enter.

Next, run the command `make dep` to update all of the source dependencies. This is an important step. `make clean` removes old binary files from the kernel source tree.

The command `make zImage` compiles the kernel and writes it to `/usr/src/linux/arch/i386/boot/zImage`. Linux kernels on Intel systems are always compressed. Sometimes the kernel you want to compile is too large to be compressed with the compression system that `make zImage` uses. A kernel which is too large will exit the kernel compile with the error message: `Kernel Image Too Large`. If this happens, try the command `make bzImage`, which uses a compression system that supports larger kernels. The kernel is written to `/usr/src/linux/arch/i386/boot/bzImage`.

Once you have the kernel compiled, you need to either copy it to a boot floppy (with a command like ``cp zImage /dev/fd0") or install the image so LILO will boot from your hard drive. See page ☐ for more information.

Grove, 4.9.1

2. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system.	Grove, as evidenced by the example citations below, discloses "said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 1.1, 1.4, and 1.5 above.	

3. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system.	Grove, as evidenced by the example citations below, discloses "said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

4. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an application program and an operating system of said computer system.	Grove, as evidenced by the example citations below, discloses "said decompressed at least a portion of said boot data comprises program code associated with an application program and an operating system of said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

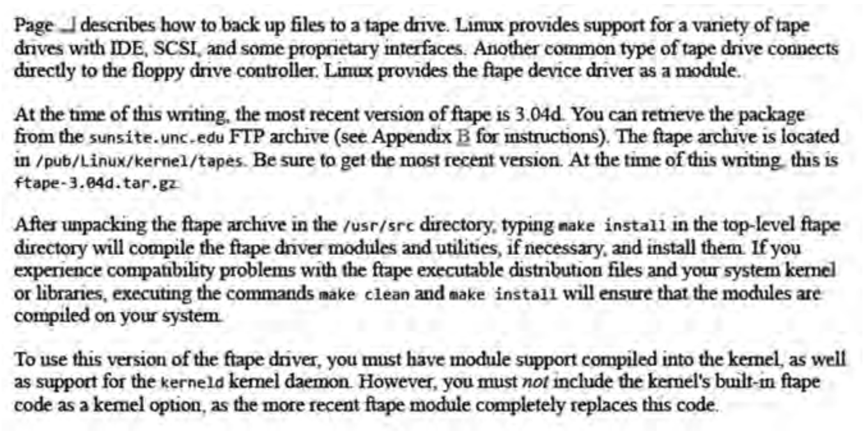
5. The method of claim 1, wherein said preloading is performed by a data storage controller connected to said boot device.	Grove, as evidenced by the example citations below, discloses "said preloading is performed by a data storage controller connected to said boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said preloading is performed by a data storage controller connected to said boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Grove discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also


Page _ describes how to back up files to a tape drive. Linux provides support for a variety of tape drives with IDE, SCSI, and some proprietary interfaces. Another common type of tape drive connects directly to the floppy drive controller. Linux provides the ftape device driver as a module.

At the time of this writing, the most recent version of ftape is 3.04d. You can retrieve the package from the `sunsite.unc.edu` FTP archive (see Appendix B for instructions). The ftape archive is located in `/pub/Linux/kernel/tapes`. Be sure to get the most recent version. At the time of this writing, this is `ftape-3.04d.tar.gz`.

After unpacking the ftape archive in the `/usr/src` directory, typing `make install` in the top-level ftape directory will compile the ftape driver modules and utilities, if necessary, and install them. If you experience compatibility problems with the ftape executable distribution files and your system kernel or libraries, executing the commands `make clean` and `make install` will ensure that the modules are compiled on your system.

To use this version of the ftape driver, you must have module support compiled into the kernel, as well as support for the `kerneld` kernel daemon. However, you must *not* include the kernel's built-in ftape code as a kernel option, as the more recent ftape module completely replaces this code.

Grove, 4.9.3

Appendix B31
Invalidity of U.S. Patent 8,090,936 based on Grove

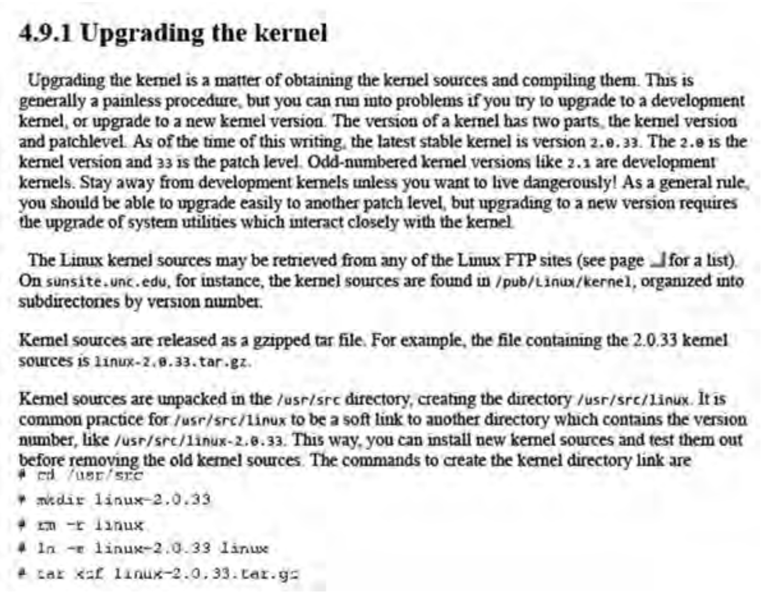
6. The method of claim 1, further comprising updating the list of boot data.	Grove, as evidenced by the example citations below, discloses "updating the list of boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Grove discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also



4.9.1 Upgrading the kernel

Upgrading the kernel is a matter of obtaining the kernel sources and compiling them. This is generally a painless procedure, but you can run into problems if you try to upgrade to a development kernel, or upgrade to a new kernel version. The version of a kernel has two parts, the kernel version and patchlevel. As of the time of this writing, the latest stable kernel is version `2.0.33`. The `2.0` is the kernel version and `33` is the patch level. Odd-numbered kernel versions like `2.1` are development kernels. Stay away from development kernels unless you want to live dangerously! As a general rule, you should be able to upgrade easily to another patch level, but upgrading to a new version requires the upgrade of system utilities which interact closely with the kernel.

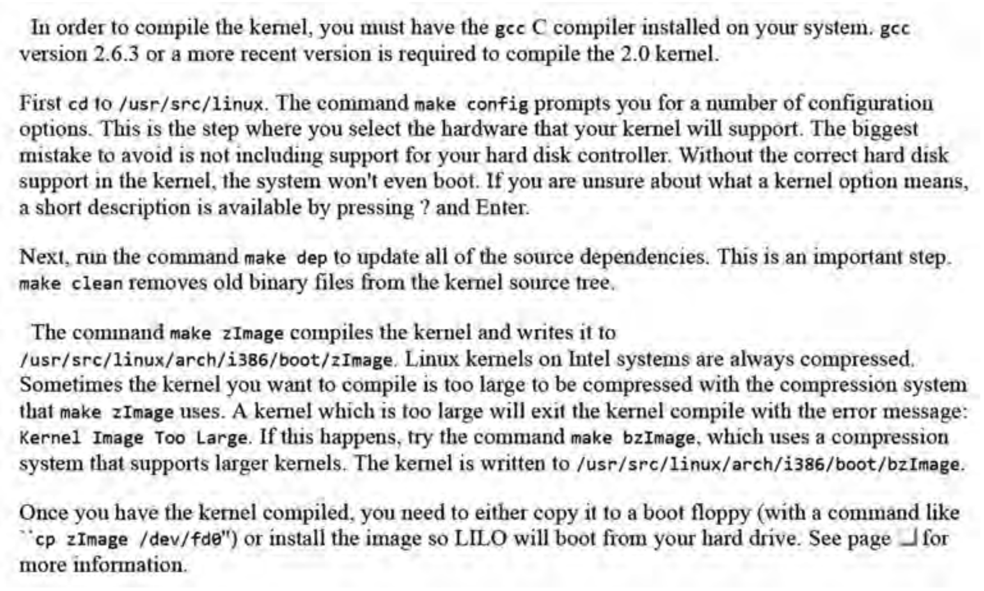
The Linux kernel sources may be retrieved from any of the Linux FTP sites (see page _ for a list). On `sunsite.unc.edu`, for instance, the kernel sources are found in `/pub/Linux/kernel`, organized into subdirectories by version number.

Kernel sources are released as a gzipped tar file. For example, the file containing the 2.0.33 kernel sources is `linux-2.0.33.tar.gz`.

Kernel sources are unpacked in the `/usr/src` directory, creating the directory `/usr/src/linux`. It is common practice for `/usr/src/linux` to be a soft link to another directory which contains the version number, like `/usr/src/linux-2.0.33`. This way, you can install new kernel sources and test them out before removing the old kernel sources. The commands to create the kernel directory link are

```
# cd /usr/src
# mkdir linux-2.0.33
# rm -r linux
# ln -s linux-2.0.33 linux
# tar xzf linux-2.0.33.tar.gz
```

Grove, 4.9.1



In order to compile the kernel, you must have the `gcc` C compiler installed on your system. `gcc` version 2.6.3 or a more recent version is required to compile the 2.0 kernel.

First `cd` to `/usr/src/linux`. The command `make config` prompts you for a number of configuration options. This is the step where you select the hardware that your kernel will support. The biggest mistake to avoid is not including support for your hard disk controller. Without the correct hard disk support in the kernel, the system won't even boot. If you are unsure about what a kernel option means, a short description is available by pressing ? and Enter.

Next, run the command `make dep` to update all of the source dependencies. This is an important step. `make clean` removes old binary files from the kernel source tree.

The command `make zImage` compiles the kernel and writes it to `/usr/src/linux/arch/i386/boot/zImage`. Linux kernels on Intel systems are always compressed. Sometimes the kernel you want to compile is too large to be compressed with the compression system that `make zImage` uses. A kernel which is too large will exit the kernel compile with the error message: `Kernel Image Too Large`. If this happens, try the command `make bzImage`, which uses a compression system that supports larger kernels. The kernel is written to `/usr/src/linux/arch/i386/boot/bzImage`.

Once you have the kernel compiled, you need to either copy it to a boot floppy (with a command like ``cp zImage /dev/fd0'') or install the image so LILO will boot from your hard drive. See page ☐ for more information.

Grove, 4.9.1

8. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form.	Grove, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

9. The method of claim 1, wherein a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form.	Grove, as evidenced by the example citations below, discloses “a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form.”

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.

Grove discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

11.1. a processor;	Grove, as evidenced by the example citations below, discloses “a processor.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Grove discloses this limitation: *See* Claim 1.2 above.	

Appendix B31
Invalidity of U.S. Patent 8,090,936 based on Grove

11.2. a memory; and	Grove, as evidenced by the example citations below, discloses "a memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 1.3, and 1.4 above.	

11.3.1 a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device,	Grove, as evidenced by the example citations below, discloses "a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 1.1, 1.3, 1.4 above.	

11.3.2 said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system; and,	Grove, as evidenced by the example citations below, discloses “said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Grove discloses this limitation: *See* Claims 1.3, 1.4, and 1.5 above.	

11.4 a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data.	Grove, as evidenced by the example citations below, discloses “a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Grove discloses this limitation: *See* Claims 1.1 and 1.5 above.	

12. The system of claim 11, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program.	Grove, as evidenced by the example citations below, discloses "said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 1.1, 3, and 11.3.1 above.	

Appendix B31
Invalidity of U.S. Patent 8,090,936 based on Grove

13. The system of claim 11, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system.	Grove, as evidenced by the example citations below, discloses "wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 1.1, 1.3, 3, 5 and 11.3.1 and 11.3.2 above.	

15. The system of claim 11, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form.	Grove, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 1, 8, and 11 above.	

16. The system of claim 11, wherein a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form.	Grove, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Grove discloses this limitation: *See* Claims 1, 9, and 11 above.	

Appendix B32
Invalidity of U.S. Patent 8,090,936 based on Jones

The publication Jones, The Microsoft Interactive TV system: An Experience Report, July 1997 ("Jones") invalidates claims 1-6, 8-9, 11-13, and 15-16 of United States Patent No. 8,090,936 ("the '936 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '936 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix B32
Invalidity of U.S. Patent 8,090,936 based on Jones

1.1 maintaining a list of boot data used for booting a computer system, wherein at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device;	Jones, as evidenced by the exemplary citations below, discloses "maintaining a list of boot data used for booting a computer system, wherein at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system, wherein at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Jones discloses this limitation:

> "The MMOSA real-time kernel was developed as a cooperative effort between members of the operating systems research group within Microsoft Research and a team of development staff committed to building a small, embedded-systems kernel meeting the needs of interactive television. MMOSA is the internal name for the ITV kernel: it stands for MultiMedia Operating System Architecture."

Jones, §4.

> "Set-top boxes boot as follows. First, a small boot loader in EPROM is run. This decompresses a boot image also in the ROM and jumps to its entry point. The ROM boot image contains the MMOSA kernel, the network protocol stack and drivers, and a network boot program. The network boot program sends a DHCP request. It waits for a DHCP reply containing not only the usual IP address, but also some MITV-specific extended results such as sets of IP addresses of Class Store servers and a boot filename to use. It then loads a new boot image from a Class Store via TFTP and jumps to it."

Jones, § 10.

> "Kanji font not in ROM. The set-top boxes we built for the NTT trial contained 1/2 MB of EPROM. This was sufficient for storing the compressed boot image but wasn't large enough to hold a Kanji font."

Jones, § 15.

1.2 initializing a central processing unit of said computer system;	Jones, as evidenced by the example citations below, discloses “initializing a central processing unit of said computer system;”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.	

1.3 preloading said at least a portion of said boot data in compressed form from said boot device to a memory;	Jones, as evidenced by the example citations below, discloses "preloading said at least a portion of said boot data in compressed form from said boot device to a memory;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading said at least a portion of said boot data in compressed form from said boot device to a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Jones discloses this limitation:

> "Set-top boxes boot as follows. First, a small boot loader in EPROM is run. This decompresses a boot image also in the ROM and jumps to its entry point. The ROM boot image contains the MMOSA kernel, the network protocol stack and drivers, and a network boot program. The network boot program sends a DHCP request. It waits for a DHCP reply containing not only the usual IP address, but also some MITV-specific extended results such as sets of IP addresses of Class Store servers and a boot filename to use. It then loads a new boot image from a Class Store via TFTP and jumps to it."

Jones, 14 at § 10.

> "Kanji font not in ROM. The set-top boxes we built for the NTT trial contained 1/2 MB of EPROM. This was sufficient for storing the compressed boot image but wasn't large enough to hold a Kanji font."

Jones, § 15.

1.4 accessing and decompressing said at least a portion of said boot data in said compressed form from said memory; and	Jones, as evidenced by the example citations below, discloses "accessing and decompressing said at least a portion of said boot data in said compressed form from said memory; and"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing and decompressing said at least a portion of said boot data in said compressed form from said memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: "Set-top boxes boot as follows. First, a small boot loader in EPROM is run. This decompresses a boot image also in the ROM and jumps to its entry point. The ROM boot image contains the MMOSA kernel, the network protocol stack and drivers, and a network boot program. The network boot program sends a DHCP request. It waits for a DHCP reply containing not only the usual IP address, but also some MITV-specific extended results such as sets of IP addresses of Class Store servers and a boot filename to use. It then loads a new boot image from a Class Store via TFTP and jumps to it." Jones, 14 at § 10. "Kanji font not in ROM. The set-top boxes we built for the NTT trial contained 1/2 MB of EPROM. This was sufficient for storing the compressed boot image but wasn't large enough to hold a Kanji font." Jones, § 15.	

1.5 utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine.	Jones, as evidenced by the example citations below, discloses "utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: "Set-top boxes boot as follows. First, a small boot loader in EPROM is run. This decompresses a boot image also in the ROM and jumps to its entry point. The ROM boot image contains the MMOSA kernel, the network protocol stack and drivers, and a network boot program. The network boot program sends a DHCP request. It waits for a DHCP reply containing not only the usual IP address, but also some MITV-specific extended results such as sets of IP addresses of Class Store servers and a boot filename to use. It then loads a new boot image from a Class Store via TFTP and jumps to it." Jones, 14 at § 10.	

2. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system.	Jones, as evidenced by the example citations below, discloses "said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claims 1.1, 1.4, and 1.5 above.	

3. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system.	Jones, as evidenced by the example citations below, discloses "said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above. *See also* "The system is designed to be able to provide traditional television service, video-on-demand service, and custom interactive applications, such as interactive shopping services, to subscriber bases scaling from a few hundred subscribers up through major metropolitan areas." Jones, Abstract. "Open software platform supporting old, new, and unforeseen applications and third-party development." Jones, §1. "From March, 1996 to April, 1997 this system served 298 paying subscribers, providing the following applications to users: • Movies on demand. The same code is used with different data to provide movies-on-demand, sports-on-demand, animation-on-demand (cartoons), and cooking-on-demand services. • Electronic program guide -- an interactive TV guide to available broadcast channels. • Viewing standard broadcast television channels. • NTT customer service application. Provides subscriber logon, channel lockout, billing queries, etc. Applications written and deployed by NTT include: • Karaoke on demand.	

> • On-line shopping services."

Jones, §2.

> "DCOM is used to implement many application services, including program invocation, remote database services (used for program and movie guide data), per-viewer and per-set-top-box persistent state (such as profile data and last channel watched), financial transaction services, and logging. Finally, on top of these lower layers, the actual user-visible applications, such as the video-on-demand player, the digital broadcast player, the electronic program guide, and the system navigator (the channel surfing application), are built."

Jones, §2.1.

> "Approximately 12 megabytes of general-purpose set-top box memory are in use when running typical applications, such as video-on-demand."

Jones, §2.1.

> "Application Portability: Applications written to the Win32 subset could be easily ported between the set-top box and existing Windows platforms. For instance, some games and Internet Explorer were both ported to the set-top box environment."

Jones, §6.1.

> "The namespace is used by client code for locating and connecting to a number of potentially replicated services at boot time and application start time."

Jones, §6.4.

4. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an application program and an operating system of said computer system.	Jones, as evidenced by the example citations below, discloses "said decompressed at least a portion of said boot data comprises program code associated with an application program and an operating system of said computer system."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Jones discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> "The system is designed to be able to provide traditional television service, video-on-demand service, and custom interactive applications, such as interactive shopping services, to subscriber bases scaling from a few hundred subscribers up through major metropolitan areas."

Jones, Abstract.

> "Open software platform supporting old, new, and unforeseen applications and third-party development."

Jones, §1.

> "From March, 1996 to April, 1997 this system served 298 paying subscribers, providing the following applications to users:
> • Movies on demand. The same code is used with different data to provide movies-on-demand, sports-on-demand, animation-on-demand (cartoons), and cooking-on-demand services.
> • Electronic program guide -- an interactive TV guide to available broadcast channels.
> • Viewing standard broadcast television channels.
> • NTT customer service application. Provides subscriber logon, channel lockout, billing queries, etc.
>
> Applications written and deployed by NTT include:

• Karaoke on demand.
• On-line shopping services."

Jones, §2.

"DCOM is used to implement many application services, including program invocation, remote database services (used for program and movie guide data), per-viewer and per-set-top-box persistent state (such as profile data and last channel watched), financial transaction services, and logging. Finally, on top of these lower layers, the actual user-visible applications, such as the video-on-demand player, the digital broadcast player, the electronic program guide, and the system navigator (the channel surfing application), are built."

Jones, §2.1.

"Approximately 12 megabytes of general-purpose set-top box memory are in use when running typical applications, such as video-on-demand."

Jones, §2.1.

"Application Portability: Applications written to the Win32 subset could be easily ported between the set-top box and existing Windows platforms. For instance, some games and Internet Explorer were both ported to the set-top box environment."

Jones, §6.1.

"The namespace is used by client code for locating and connecting to a number of potentially replicated services at boot time and application start time."

Jones, §6.4.

5. The method of claim 1, wherein said preloading is performed by a data storage controller connected to said boot device.	Jones, as evidenced by the example citations below, discloses "said preloading is performed by a data storage controller connected to said boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said preloading is performed by a data storage controller connected to said boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above. *See also* "For instance the Tiger controller server runs on a dedicated machine, and the Tiger cubs (the actual server machines from which striped video is delivered) occupy ten dedicated server machines, leaving four other general head-end server machines in this configuration." Jones, §2.2. *See also* Jones, Fig 2-1. "The set-top box also has a bi-directional infrared port for communicating with the hand controller, a smart card interface, a serial port (used for debugging), a microphone input, auxiliary audio & video inputs, and separate audio and video outputs for TV and VCR." Jones, §3. "The Tiger system deployed in the NTT trial uses one dedicated Tiger controller server and ten dedicated Tiger Cub servers (the machines across which the data are striped)." Jones, §8.1.	

6. The method of claim 1, further comprising updating the list of boot data.	Jones, as evidenced by the example citations below, discloses "updating the list of boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

8. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form.	Jones, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

9. The method of claim 1, wherein a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form.	Jones, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Jones discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> "Custom video and audio hardware for the set-top box contains a MPEG-2 decoder, NTSC (the U.S. and Japanese analog television encoding standard) encoders & decoders, a tuner, and an audio mixer."

Jones, §3.

> "Everything from server machines and ATM switches to real-time MPEG-2 encoders and OS software was precisely duplicated."

Jones, §11.

11.1. a processor;	Jones, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claim 1.2 above.	

11.2. a memory; and	Jones, as evidenced by the example citations below, discloses “a memory.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Jones discloses this limitation: *See* Claims 1.3, and 1.4 above.	

11.3.1 a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device,	Jones, as evidenced by the example citations below, discloses "a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Jones discloses this limitation:

See Claims 1.1, 1.3, 1.4 above.

See also

> "Approximately 12 megabytes of general-purpose set-top box memory are in use when running typical applications, such as video-on-demand. This includes 3MB of Kanji font memory and 3MB of video buffer memory. In addition to the general-purpose memory, 1.7 megabytes of special video memory and 2 megabytes of dedicated MPEG-2 decode memory are in use."

Jones, §2.1.

> "Set-Top Box Memory Usage."

Jones, §12.4. See generally, Jones §12.4

> "• Kanji font not in ROM. The set-top boxes we built for the NTT trial contained 1/2 MB of EPROM. This was sufficient for storing the compressed boot image but wasn't large enough to hold a Kanji font. So the font occupied about 3MB of RAM. Unfortunately, due to lack of

address lines, the ROM size could not be increased.

• Space cost of code reuse and time pressures. As discussed in section 12.4, the set-top box memory budget was exceeded in a number of ways. Prime contributors were the video buffering granularity used, the Kanji font problem, the inclusion of DCE RPC, use of the Windows NT driver model, and use of the Microsoft Foundation Classes."

Jones, §15.

11.3.2 said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system; and,	Jones, as evidenced by the example citations below, discloses "said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claims 1.3, 1.4, and 1.5 above.	

11.4 a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data.	Jones, as evidenced by the example citations below, discloses "a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claims 1.1 and 1.5 above.	

12. The system of claim 11, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program.	Jones, as evidenced by the example citations below, discloses "said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claims 1.1, 3, and 11.3.1 above.	

13. The system of claim 11, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system.	Jones, as evidenced by the example citations below, discloses "wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claims 1.1, 1.3, 3, 5 and 11.3.1 and 11.3.2 above.	

15. The system of claim 11, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form.	Jones, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Jones discloses this limitation: *See* Claims 1, 8, and 11 above.	

16. The system of claim 11, wherein a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form.	Jones, as evidenced by the example citations below, discloses “a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Jones discloses this limitation: *See* Claims 1, 9, and 11 above.	

Appendix B33
Invalidity of U.S. Patent 8,090,936 based on Magstar

The publication Magstar and IBM 3590 High Performance Tape Subsystem Technical Guide, November 1996 ("Magstar") invalidates claims 1-6, 8-9, 11-13, and 15-16 of United States Patent No. 8,090,936 ("the '936 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '936 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix B33
Invalidity of U.S. Patent 8,090,936 based on Magstar

1.1 maintaining a list of boot data used for booting a computer system, wherein at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device;	Magstar, as evidenced by the exemplary citations below, discloses "maintaining a list of boot data used for booting a computer system, wherein at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system, wherein at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Magstar discloses this limitation:

> "The IBMLZ1 compression algorithm used in the IBM 3590 is based on the Ziv-Lempel algorithm2 and Jackson's[3] class of encoding methods."

Magstar, 25. *See also generally* Magstar, 25-26 at "Improved compression."

> "It is during the process of transferring data from the host channel to the buffer that the data can be compressed using the BAC algorithm component of IDRC."

Magstar, 132.

> "The compression achieved using IDRC is made up of two factors:
>
> - Automatic reblocking of the data to a block size of 128KB, which has a more marked effect on small block sizes
> - Applying the BAC algorithm to the data, the effectiveness of which depends on the randomness of the data, and whether or not it has already been compressed (for example, by hardware or software data compression in the host)."

Magstar, 132. *See also generally* Magstar, 132 at § 6.1.2 and §6.9.

> "New longitudinal technology now announced by IBM (16-track

Serpentine Interleaved Longitudinal Recording) significantly improves tape performance and transfer rates without changing the tape speed (2 m/s). A buffer is used and the data compressed before it is written to.” Magstar, 12.

1.2 initializing a central processing unit of said computer system;	Magstar, as evidenced by the example citations below, discloses "initializing a central processing unit of said computer system;"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

1.3 preloading said at least a portion of said boot data in compressed form from said boot device to a memory;	Magstar, as evidenced by the example citations below, discloses "preloading said at least a portion of said boot data in compressed form from said boot device to a memory;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading said at least a portion of said boot data in compressed form from said boot device to a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Magstar discloses this limitation:

> "The IBMLZ1 compression algorithm used in the IBM 3590 is based on the Ziv-Lempel algorithm2 and Jackson's[3] class of encoding methods."

Magstar, 25. *See also generally* Magstar, 25-26 at "Improved compression."

> "It is during the process of transferring data from the host channel to the buffer that the data can be compressed using the BAC algorithm component of IDRC."

Magstar, 132.

> "The compression achieved using IDRC is made up of two factors:
>
> - Automatic reblocking of the data to a block size of 128KB, which has a more marked effect on small block sizes
> - Applying the BAC algorithm to the data, the effectiveness of which depends on the randomness of the data, and whether or not it has already been compressed (for example, by hardware or software data compression in the host)."

Magstar, 132. *See also generally* Magstar, 132 at § 6.1.2 and §6.9.

> "New longitudinal technology now announced by IBM (16-track Serpentine Interleaved Longitudinal Recording) significantly improves tape performance and transfer rates without changing the tape speed (2 m/s). A buffer is used and the data compressed before it is written to."

Magstar, 12.

1.4 accessing and decompressing said at least a portion of said boot data in said compressed form from said memory; and	Magstar, as evidenced by the example citations below, discloses "accessing and decompressing said at least a portion of said boot data in said compressed form from said memory; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing and decompressing said at least a portion of said boot data in said compressed form from said memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Magstar discloses this limitation:

> "It is during the process of transferring data from the host channel to the buffer that the data can be compressed using the BAC algorithm component of IDRC."

Magstar, 132.

> "The compression achieved using IDRC is made up of two factors:
>
> - Automatic reblocking of the data to a block size of 128KB, which has a more marked effect on small block sizes
> - Applying the BAC algorithm to the data, the effectiveness of which depends on the randomness of the data, and whether or not it has already been compressed (for example, by hardware or software data compression in the host)."

Magstar, 132. *See also generally* Magstar, 132 at § 6.1.2 and §6.9.

> "New longitudinal technology now announced by IBM (16-track Serpentine Interleaved Longitudinal Recording) significantly improves tape performance and transfer rates without changing the tape speed (2 m/s). A buffer is used and the data compressed before it is written to."

Magstar, 12.

1.5 utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine.	Magstar, as evidenced by the example citations below, discloses "utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Magstar discloses this limitation: "The IBMLZ1 compression algorithm used in the IBM 3590 is based on the Ziv-Lempel algorithm2 and Jackson's[3] class of encoding methods." Magstar, 25. *See also generally* Magstar, 25-26 at "Improved compression." "New longitudinal technology now announced by IBM (16-track Serpentine Interleaved Longitudinal Recording) significantly improves tape performance and transfer rates without changing the tape speed (2 m/s). A buffer is used and the data compressed before it is written to." Magstar, 12.	

2. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system.	Magstar, as evidenced by the example citations below, discloses “said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Magstar discloses this limitation: *See* Claims 1.1, 1.4, and 1.5 above.	

3. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system.	Magstar, as evidenced by the example citations below, discloses "said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Magstar discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> "This support is functionally transparent to the user programs with the exception of those applications that use the NOTE TYPE=ABS macro and calculate the logical block number on the basis of the values returned in register 0 and resister 1. These application programs may have to be modified."

Magstar, 105.

> "Typically the user application program itself does not use the tape drive directly."

Magstar, 122.

4. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an application program and an operating system of said computer system.	Magstar, as evidenced by the example citations below, discloses "said decompressed at least a portion of said boot data comprises program code associated with an application program and an operating system of said computer system."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Magstar discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> "This support is functionally transparent to the user programs with the exception of those applications that use the NOTE TYPE=ABS macro and calculate the logical block number on the basis of the values returned in register 0 and resister 1. These application programs may have to be modified."

Magstar, 105.

> "Typically the user application program itself does not use the tape drive directly."

Magstar, 122.

5. The method of claim 1, wherein said preloading is performed by a data storage controller connected to said boot device.	Magstar, as evidenced by the example citations below, discloses "said preloading is performed by a data storage controller connected to said boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said preloading is performed by a data storage controller connected to said boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Magstar discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> "To use the Magstar tape drive from an ES/9000 or S/390 system, an IBM 3590-A00 tape controller is required to convert the channel commands to corresponding SCSI commands."

Magstar, 96.

> "The IBM Magstar Virtual Tape Server Controller, Tape Volume Cache, and the IBM 3590 Magstar tape drives, together with the required housing, make up the IBM Magstar Virtual Tape Server (VTS) subsystem, allowing automatic utilization of the Magstar cartridge storage capacity and the drive data rate of 9 MB/s."

Magstar, 150.

> "The IBM Magstar Tape Server Controller and its associated microcode, are the key components of the Virtual Tape Server subsystem:
>
> • It automatically fills and manages Magstar 3590 cartridge capacity.
>
> • It controls and manages the volume movement to and from the tape volume cache and 3590 cartridges or Magstar tape drives."

Magstar, 150. *See also generally*, Magstar 1-269.

6. The method of claim 1, further comprising updating the list of boot data.	Magstar, as evidenced by the example citations below, discloses "updating the list of boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Magstar discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

8. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form.	Magstar, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Magstar discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> "The IBMLZ1 compression algorithm used in the IBM 3590 is based on the Ziv-Lempel algorithm2 and Jackson's[3] class of encoding methods."

Magstar, 25. *See also generally* Magstar, 25-26 at "Improved compression."

> "It is during the process of transferring data from the host channel to the buffer that the data can be compressed using the BAC algorithm component of IDRC."

Magstar, 132.

> "The compression achieved using IDRC is made up of two factors:
>
> - Automatic reblocking of the data to a block size of 128KB, which has a more marked effect on small block sizes
> - Applying the BAC algorithm to the data, the effectiveness of which depends on the randomness of the data, and whether or not it has already been compressed (for example, by hardware or software data compression in the host)."

Magstar, 132. *See also generally* Magstar, 132 at § 6.1.2 and §6.9.

> "New longitudinal technology now announced by IBM (16-track Serpentine Interleaved Longitudinal Recording) significantly improves tape performance and transfer rates without changing the tape speed (2

m/s). A buffer is used and the data compressed before it is written to.” Magstar, 12.

9. The method of claim 1, wherein a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form.	Magstar, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Magstar discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> "The IBMLZ1 compression algorithm used in the IBM 3590 is based on the Ziv-Lempel algorithm2 and Jackson's3 class of encoding methods."

Magstar, 25.

11.1. a processor;	Magstar, as evidenced by the example citations below, discloses “a processor.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Magstar discloses this limitation: *See* Claim 1.2 above.	

11.2. a memory; and	Magstar, as evidenced by the example citations below, discloses “a memory.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Magstar discloses this limitation: *See* Claims 1.3, and 1.4 above.	

11.3.1 a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device,	Magstar, as evidenced by the example citations below, discloses "a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Magstar discloses this limitation: *See* Claims 1.1, 1.3, 1.4 above.	

11.3.2 said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system; and,	Magstar, as evidenced by the example citations below, discloses "said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Magstar discloses this limitation: *See* Claims 1.3, 1.4, and 1.5 above.	

11.4 a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data.	Magstar, as evidenced by the example citations below, discloses "a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Magstar discloses this limitation: *See* Claims 1.1 and 1.5 above.	

12. The system of claim 11, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program.	Magstar, as evidenced by the example citations below, discloses "said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Magstar discloses this limitation: *See* Claims 1.1, 3, and 11.3.1 above.	

13. The system of claim 11, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system.	Magstar, as evidenced by the example citations below, discloses "wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Magstar discloses this limitation: *See* Claims 1.1, 1.3, 3, 5 and 11.3.1 and 11.3.2 above.	

15. The system of claim 11, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form.	Magstar, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Magstar discloses this limitation: *See* Claims 1, 8, and 11 above.	

16. The system of claim 11, wherein a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form.	Magstar, as evidenced by the example citations below, discloses “a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Magstar discloses this limitation: *See* Claims 1, 9, and 11 above.	

Appendix B34
Invalidity of U.S. Patent 8,090,936 based on Mealey

The publication Mealey, B, IBM, An IP.com Prior Art Database Technical Disclosure, January, 1992 ("Mealey") invalidates claims 1-6, 8-9, 11-13, and 15-16 of United States Patent No. 8,090,936 ("the '936 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '936 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix B34
Invalidity of U.S. Patent 8,090,936 based on Mealey

1.1 maintaining a list of boot data used for booting a computer system, wherein at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device;	Mealey, as evidenced by the exemplary citations below, discloses "maintaining a list of boot data used for booting a computer system, wherein at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system, wherein at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Mealey discloses this limitation:

> "Disclosed is an approach that reduces resources and time required for system initialization. This is accomplished by applying a data compression algorithm to the text and data of a boot-image.
> The boot-image is divided into two portions: text and data. The text is compressed and bound with a decompression program. It is decompressed immediately after being loaded.
> The data portion is a RAM disk file system. The file system is compressed at the block level. It is accessed by a pseudo device driver that compresses on writes and decompresses on reads. The file system remains compressed during system initialization reducing the amount of memory required to hold it.
> Applying compression to a boot image can produce a significant time savings when booting from a slow device, such as a busy LAN or floppy disk. It also reduces the size of the media required to hold the image. Finally, it can reduce the amount of memory required to hold the boot image until system initialization is complete."

Mealey, 1.

1.2 initializing a central processing unit of said computer system;	Mealey, as evidenced by the example citations below, discloses "initializing a central processing unit of said computer system;"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Mealey discloses this limitation: "Disclosed is an approach that reduces resources and time required for system initialization. This is accomplished by applying a data compression algorithm to the text and data of a boot-image. The boot-image is divided into two portions: text and data. The text is compressed and bound with a decompression program. It is decompressed immediately after being loaded. The data portion is a RAM disk file system. The file system is compressed at the block level. It is accessed by a pseudo device driver that compresses on writes and decompresses on reads. The file system remains compressed during system initialization reducing the amount of memory required to hold it. Applying compression to a boot image can produce a significant time savings when booting from a slow device, such as a busy LAN or floppy disk. It also reduces the size of the media required to hold the image. Finally, it can reduce the amount of memory required to hold the boot image until system initialization is complete." Mealey, 1.	

1.3 preloading said at least a portion of said boot data in compressed form from said boot device to a memory;	Mealey, as evidenced by the example citations below, discloses "preloading said at least a portion of said boot data in compressed form from said boot device to a memory;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading said at least a portion of said boot data in compressed form from said boot device to a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Mealey discloses this limitation:

> "Disclosed is an approach that reduces resources and time required for system initialization. This is accomplished by applying a data compression algorithm to the text and data of a boot-image.
> The boot-image is divided into two portions: text and data. The text is compressed and bound with a decompression program. It is decompressed immediately after being loaded.
> The data portion is a RAM disk file system. The file system is compressed at the block level. It is accessed by a pseudo device driver that compresses on writes and decompresses on reads. The file system remains compressed during system initialization reducing the amount of memory required to hold it.
> Applying compression to a boot image can produce a significant time savings when booting from a slow device, such as a busy LAN or floppy disk. It also reduces the size of the media required to hold the image. Finally, it can reduce the amount of memory required to hold the boot image until system initialization is complete."

Mealey, 1.

1.4 accessing and decompressing said at least a portion of said boot data in said compressed form from said memory; and	Mealey, as evidenced by the example citations below, discloses "accessing and decompressing said at least a portion of said boot data in said compressed form from said memory; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing and decompressing said at least a portion of said boot data in said compressed form from said memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Mealey discloses this limitation:

> "Disclosed is an approach that reduces resources and time required for system initialization. This is accomplished by applying a data compression algorithm to the text and data of a boot-image.
> The boot-image is divided into two portions: text and data. The text is compressed and bound with a decompression program. It is decompressed immediately after being loaded.
> The data portion is a RAM disk file system. The file system is compressed at the block level. It is accessed by a pseudo device driver that compresses on writes and decompresses on reads. The file system remains compressed during system initialization reducing the amount of memory required to hold it.
> Applying compression to a boot image can produce a significant time savings when booting from a slow device, such as a busy LAN or floppy disk. It also reduces the size of the media required to hold the image. Finally, it can reduce the amount of memory required to hold the boot image until system initialization is complete."

Mealey, 1.

1.5 utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine.	Mealey, as evidenced by the example citations below, discloses "utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Mealey discloses this limitation: "Disclosed is an approach that reduces resources and time required for system initialization. This is accomplished by applying a data compression algorithm to the text and data of a boot-image. The boot-image is divided into two portions: text and data. The text is compressed and bound with a decompression program. It is decompressed immediately after being loaded. The data portion is a RAM disk file system. The file system is compressed at the block level. It is accessed by a pseudo device driver that compresses on writes and decompresses on reads. The file system remains compressed during system initialization reducing the amount of memory required to hold it. Applying compression to a boot image can produce a significant time savings when booting from a slow device, such as a busy LAN or floppy disk. It also reduces the size of the media required to hold the image. Finally, it can reduce the amount of memory required to hold the boot image until system initialization is complete." Mealey, 1.	

2. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system.	Mealey, as evidenced by the example citations below, discloses "said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Mealey discloses this limitation: *See* Claims 1.1, 1.4, and 1.5 above.	

3. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system.	Mealey, as evidenced by the example citations below, discloses "said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Mealey discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

4. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an application program and an operating system of said computer system.	Mealey, as evidenced by the example citations below, discloses “said decompressed at least a portion of said boot data comprises program code associated with an application program and an operating system of said computer system.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Mealey discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

5. The method of claim 1, wherein said preloading is performed by a data storage controller connected to said boot device.	Mealey, as evidenced by the example citations below, discloses "said preloading is performed by a data storage controller connected to said boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said preloading is performed by a data storage controller connected to said boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Mealey discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

6. The method of claim 1, further comprising updating the list of boot data.	Mealey, as evidenced by the example citations below, discloses "updating the list of boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Mealey discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

8. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form.	Mealey, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Mealey discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

9. The method of claim 1, wherein a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form.	Mealey, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Mealey discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

11.1. a processor;	Mealey, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Mealey discloses this limitation: *See* Claim 1.2 above.	

11.2. a memory; and	Mealey, as evidenced by the example citations below, discloses “a memory.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Mealey discloses this limitation: *See* Claims 1.3, and 1.4 above.	

11.3.1 a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device,	Mealey, as evidenced by the example citations below, discloses "a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Mealey discloses this limitation: *See* Claims 1.1, 1.3, 1.4 above.	

11.3.2 said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system; and,	Mealey, as evidenced by the example citations below, discloses "said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Mealey discloses this limitation: *See* Claims 1.3, 1.4, and 1.5 above.	

11.4 a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data.	Mealey, as evidenced by the example citations below, discloses “a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Mealey discloses this limitation: *See* Claims 1.1 and 1.5 above.	

12. The system of claim 11, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program.	Mealey, as evidenced by the example citations below, discloses "said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Mealey discloses this limitation: *See* Claims 1.1, 3, and 11.3.1 above.	

13. The system of claim 11, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system.	Mealey, as evidenced by the example citations below, discloses "wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Mealey discloses this limitation: *See* Claims 1.1, 1.3, 3, 5 and 11.3.1 and 11.3.2 above.	

15. The system of claim 11, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form.	Mealey, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Mealey discloses this limitation: *See* Claims 1, 8, and 11 above.	

16. The system of claim 11, wherein a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form.	Mealey, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Mealey discloses this limitation: *See* Claims 1, 9, and 11 above.	

Appendix B35
Invalidity of U.S. Patent 8,090,936 based on Menon

The publication Menon, A performance comparison of RAID-5 and log-structured arrays, IBM Almaden Research Center, ("Menon") invalidates claims 1-6, 8-9, 11-13, and 15-16 of United States Patent No. 8,090,936 ("the '936 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '936 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix B35
Invalidity of U.S. Patent 8,090,936 based on Menon

1.1 maintaining a list of boot data used for booting a computer system, wherein at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device;	Menon, as evidenced by the exemplary citations below, discloses "maintaining a list of boot data used for booting a computer system, wherein at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device;"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system, wherein at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Menon discloses this limitation: "The record is compressed as soon as it reaches the subsystem. Compressed records are stored in the controller cache." Menon, 169. "In LSA, data is stored on disks in compressed form. After a piece of data is updated, it may not compress as well as it did before it was updated, so it may not fit back into the space that had been allocated for it bcfore the update." Menon, 169.	

Appendix B35
Invalidity of U.S. Patent 8,090,936 based on Menon

1.2 initializing a central processing unit of said computer system;	Menon, as evidenced by the example citations below, discloses "initializing a central processing unit of said computer system;"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

1.3 preloading said at least a portion of said boot data in compressed form from said boot device to a memory;	Menon, as evidenced by the example citations below, discloses "preloading said at least a portion of said boot data in compressed form from said boot device to a memory;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading said at least a portion of said boot data in compressed form from said boot device to a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Menon discloses this limitation:

> "In addition to improved transfer times, performance benefits result from the fact that by storing compressed data in the subsystem cache, we get an effectively larger cache."

Menon, 167.

> "In LSA, data is stored on disks in compressed form. After a piece of data is updated, it may not compress as well as it did before it was updated, so it may not fit back into the space that had been allocated for it bcfore the update."

Menon, 169.

> "The record is compressed as soon as it reaches the subsystem. Compressed records are stored in the controller cache."

Menon, 169.

> "When the host system tries to read a record, we fetch the entire logical track containing that record from disk. This entire track is stored in the controller cache; the requested record alone is decompressed and sent to the host."

Menon, 169.

1.4 accessing and decompressing said at least a portion of said boot data in said compressed form from said memory; and	Menon, as evidenced by the example citations below, discloses "accessing and decompressing said at least a portion of said boot data in said compressed form from said memory; and"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing and decompressing said at least a portion of said boot data in said compressed form from said memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Menon discloses this limitation: "When the host system tries to read a record, we fetch the entire logical track containing that record from disk. This entire track is stored in the controller cache; the requested record alone is decompressed and sent to the host." Menon, 169.	

1.5 utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine.	Menon, as evidenced by the example citations below, discloses "utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Menon discloses this limitation: "When the host system tries to read a record, we fetch the entire logical track containing that record from disk. This entire track is stored in the controller cache; the requested record alone is decompressed and sent to the host." Menon, 169.	

2. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system.	Menon, as evidenced by the example citations below, discloses “said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Menon discloses this limitation: *See* Claims 1.1, 1.4, and 1.5 above.	

3. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system.	Menon, as evidenced by the example citations below, discloses "said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Menon discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

4. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an application program and an operating system of said computer system.	Menon, as evidenced by the example citations below, discloses “said decompressed at least a portion of said boot data comprises program code associated with an application program and an operating system of said computer system.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Menon discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

5. The method of claim 1, wherein said preloading is performed by a data storage controller connected to said boot device.	Menon, as evidenced by the example citations below, discloses "said preloading is performed by a data storage controller connected to said boot device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said preloading is performed by a data storage controller connected to said boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Menon discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> "In this paper, we compare the performance of the well-known RAIDS arrays to that of log-structured arrays (LSA), on transaction-processing workloads. LSA borrows heavily from the log-structured file system (LFS) approach, but is executed in an outboard disk controller."

Menon, 167.

> "The RAID controller is assumed to have a read and write cache built from Non-Volatile Storage (NVS)."

Menon, 167.

> "The array uses Fast Write; When a disk block to be written is received, the block is first stored in 2 separate NVS memory locations in the array controller (to avoid single points of failure). At this point, the disk array controller signals successful completion of the write to the host. Disk blocks in array controller cache memory that need to be written to disk are called dirty. Disk blocks in cache memory that are identical to their counterparts on disk are called clean."

Menon, 167. *See also* Menon, 167-167 (Analysis of Cached RAIDs), 168 (Disk, Controller, and Channel Parameters).

> "When the host system tries to read a record, we fetch the entire logical track containing that record from disk. This entire track is stored in the

> controller cache; the requested record alone is decompressed and sent to the host."

Menon, 169.

> "The memory segment is a section of controller memory, logically organized as N + 1 segment-columns called memory segment -columns; N data memory segment-columns and 1 parity memory segment-column."

Menon, 170.

> "As for RAID-5, we assume that the controller consists of multiple (p) processors."

Menon, 170.

6. The method of claim 1, further comprising updating the list of boot data.	Menon, as evidenced by the example citations below, discloses "updating the list of boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Menon discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

8. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form.	Menon, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Menon discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

9. The method of claim 1, wherein a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form.	Menon, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Menon discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

11.1. a processor;	Menon, as evidenced by the example citations below, discloses “a processor.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Menon discloses this limitation: *See* Claim 1.2 above.	

11.2. a memory; and	Menon, as evidenced by the example citations below, discloses "a memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Menon discloses this limitation: *See* Claims 1.3, and 1.4 above.	

11.3.1 a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device,	Menon, as evidenced by the example citations below, discloses "a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Menon discloses this limitation:

See Claims 1.1, 1.3, 1.4 above.

See also

> "The LSA technique we examine combines LFS, RAID, compression and Non-Volatile cache."

Menon 167.

> "We begin by analyzing the performance of RAID-5 arrays with Non-Volatile Caching using an analytical model."

Menon, 167.

> "The RAID controller is assumed to have a read and write cache built from Non-Volatile Storage (NVS)."

Menon, 167.

> "We assume a Non-Volatile cache and the use of Fast Write (as for RAID-5)."

Menon, 169. "On a write hit or miss, the data block is accepted from the system and placed in the Non-Volatile cache after which the write is considered 'done"." Menon, 170.

11.3.2 said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system; and,	Menon, as evidenced by the example citations below, discloses "said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Menon discloses this limitation: *See* Claims 1.3, 1.4, and 1.5 above.	

11.4 a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data.	Menon, as evidenced by the example citations below, discloses "a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Menon discloses this limitation: *See* Claims 1.1 and 1.5 above.	

12. The system of claim 11, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program.	Menon, as evidenced by the example citations below, discloses "said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Menon discloses this limitation: *See* Claims 1.1, 3, and 11.3.1 above.	

13. The system of claim 11, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system.	Menon, as evidenced by the example citations below, discloses "wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Menon discloses this limitation: *See* Claims 1.1, 1.3, 3, 5 and 11.3.1 and 11.3.2 above.	

15. The system of claim 11, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form.	Menon, as evidenced by the example citations below, discloses “Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Menon discloses this limitation: *See* Claims 1, 8, and 11 above.	

16. The system of claim 11, wherein a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form.	Menon, as evidenced by the example citations below, discloses “a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Menon discloses this limitation: *See* Claims 1, 9, and 11 above.	

Appendix B36
Invalidity of U.S. Patent 8,090,936 based on Rubini

The publication Rubini, Booting the Kernel, Linux Journal, Jan. 1997, ("Rubini") invalidates claims 1-6, 8-9, 11-13, and 15-16 of United States Patent No. 8,090,936 ("the '936 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '936 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix B36
Invalidity of U.S. Patent 8,090,936 based on Rubini

1.1 maintaining a list of boot data used for booting a computer system, wherein at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device;	Rubini, as evidenced by the exemplary citations below, discloses "maintaining a list of boot data used for booting a computer system, wherein at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system, wherein at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Rubini discloses this limitation:

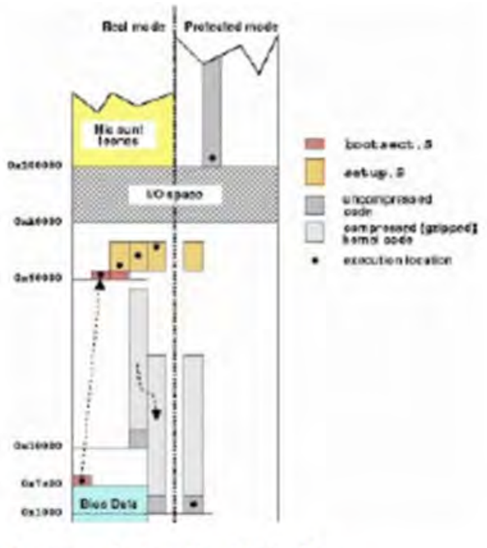


Figure 1. System Boot Data Map

Fig. 1.

> "In order to be able to use the computer when the power is turned on, the processor begins execution from the system's firmware. The firmware is "unmovable software" found in ROM; some manufacturers call it the Basic Input-Output System (BIOS) to underline its software role, some call it PROM or "flash" to stress its hardware implementation, while

others call it “console” to focus on user interaction.

The firmware usually checks the hardware's functionality, retrieves part (or all) of the kernel from a storage medium and executes it. This first part of the kernel must load the rest of itself and initialize the whole system.”

Rubini, 1.

“The file called zImage is the compressed kernel image that resides in arch/i386/boot after either make zImage or make boot is executed—the latter invocation is the one I prefer, as it works unchanged on other platforms. If you built a “big zImage” instead, the kernel file created is called bzImage and resides in the same directory.”

Rubini, 2.

“The boot steps shown above rely on the assumption that the compressed kernel can fit in half a megabyte of space. While this is true most of the time, a system stuffed with device drivers might not fit into this space. For example, kernels used in installation disks can easily outgrow the available space. Some new method is needed to solve the problem—this new method is called bzImage and was introduced in kernel version 1.3.73.

A bzImage is generated by issuing make bzImage from the top level Linux source directory. This kind of kernel image boots similarly to zImage, with a few changes:

- When the system is loaded to address 0x10000, a little helper routine is called after loading each 64K data block. The helper routine moves the data block to high memory by using a special BIOS call. Only the newer BIOS versions implement this functionality, and so, make boot still builds the conventional zImage, though this may change in the near future.
- setup.S doesn't move the system back to 0x1000 (4K) but, after entering protected mode, jumps instead directly to address 0x100000 (1MB) where data has been moved by the BIOS in the previous step.”

Rubini, 3-4.

“The rule for building the big compressed image can be read from Makefile; it affects several files in arch/i386/boot. One good point of bzImage is that when kernel/head.S is called, it doesn't notice the extra

work, and everything goes forward as usual.” Rubini, 4.

Appendix B36
Invalidity of U.S. Patent 8,090,936 based on Rubini

1.2 initializing a central processing unit of said computer system;	Rubini, as evidenced by the example citations below, discloses "initializing a central processing unit of said computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Rubini discloses this limitation:

> "In order to be able to use the computer when the power is turned on, the processor begins execution from the system's firmware. The firmware is "unmovable software" found in ROM; some manufacturers call it the Basic Input-Output System (BIOS) to underline its software role, some call it PROM or "flash" to stress its hardware implementation, while others call it "console" to focus on user interaction.
>
> The firmware usually checks the hardware's functionality, retrieves part (or all) of the kernel from a storage medium and executes it. This first part of the kernel must load the rest of itself and initialize the whole system."

Rubini, 1.

> "When the x86 processor is turned on, it is a 16-bit processor that sees only 1MB of RAM. This environment is known as "real mode" and is dictated by compatibility with older processors of the same family."

Rubini, 1.

> "Then code at 0x90200 (defined in setup.S) takes care of some hardware initialization and allows the default text mode (video.S) to be changed. Text mode selection is a compile-time option from 2.1.9 onwards."

Rubini, 2.

1.3 preloading said at least a portion of said boot data in compressed form from said boot device to a memory;	Rubini, as evidenced by the example citations below, discloses "preloading said at least a portion of said boot data in compressed form from said boot device to a memory;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading said at least a portion of said boot data in compressed form from said boot device to a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Rubini discloses this limitation:

> The firmware usually checks the hardware's functionality, retrieves part (or all) of the kernel from a storage medium and executes it. This first part of the kernel must load the rest of itself and initialize the whole system."

Rubini, 1.

> "When the x86 processor is turned on, it is a 16-bit processor that sees only 1MB of RAM. This environment is known as "real mode" and is dictated by compatibility with older processors of the same family."

Rubini, 1.

> "To make things difficult, the PC firmware loads only half a kilobyte of code and establishes its own memory layout before loading this first sector. Whatever the boot media, the first sector of the boot partition is loaded into memory at the address 0x7c00, where execution begins. What happens at 0x7c00 depends on the boot loader being used; we examine three situations here: no boot-loader, LILO, Loadlin."

Rubini, 2.

> "Booting zImage and bzImage
>
> Even though it's rare to boot the system without a boot loader, it is still possible to do so by copying the raw kernel to a floppy disk. The command cat zImage >/dev/fd0 works perfectly on Linux, although some other Unix systems can do the task reliably only by using the dd command. Without going into detail, the raw floppy image created by

zImage can then be configured using the rdev program.

The file called zImage is the compressed kernel image that resides in arch/i386/boot after either make zImage or make boot is executed—the latter invocation is the one I prefer, as it works unchanged on other platforms. If you built a "big zImage" instead, the kernel file created is called bzImage and resides in the same directory.

Booting an x86 kernel is a tricky task because of the limited amount of available memory. The Linux kernel tries to maximize usage of the low 640 kilobytes by moving itself around several times. Let's look at the steps performed by a zImage kernel in detail; all of the following path names are relative to the arch/i386/boot directory.

- The first sector (executing at 0x7c00) moves itself to 0x90000 and loads subsequent sectors after itself, getting them from the boot device using the firmware's functions to access the disk. The rest of the kernel is then loaded to address 0x10000, allowing for a maximum size of half a megabyte of data—remember, this is the compressed image. The boot sector code lives in bootsect.S, a real-mode assembly file.
- Then code at 0x90200 (defined in setup.S) takes care of some hardware initialization and allows the default text mode (video.S) to be changed. Text mode selection is a compile-time option from 2.1.9 onwards.
- Later, all the kernel is moved from 0x10000 (64K) to 0x1000 (4K). This move overwrites BIOS data stored in RAM, so BIOS calls can no longer be performed. The first physical page is not touched because it is the so-called "zero-page", used in handling virtual memory.
- At this point, setup.S enters protected mode and jumps to 0x1000, where the kernel lives. All the available memory can be accessed now, and the system can begin to run."

Rubini, 2-3.

"The boot steps shown above rely on the assumption that the compressed kernel can fit in half a megabyte of space. While this is true most of the time, a system stuffed with device drivers might not fit into this space. For example, kernels used in installation disks can easily outgrow the available space. Some new method is needed to solve the problem—this new method is called bzImage and was introduced in kernel version 1.3.73.

A bzImage is generated by issuing make bzImage from the top level Linux source directory. This kind of kernel image boots similarly to zImage, with a few changes:

- When the system is loaded to address 0x10000, a little helper routine is called after loading each 64K data block. The helper routine moves the data block to high memory by using a special BIOS call. Only the newer BIOS versions implement this functionality, and so, make boot still builds the conventional zImage, though this may change in the near future.
- setup.S doesn't move the system back to 0x1000 (4K) but, after entering protected mode, jumps instead directly to address 0x100000 (1MB) where data has been moved by the BIOS in the previous step."

Rubini, 3-4.

> "The rule for building the big compressed image can be read from Makefile; it affects several files in arch/i386/boot. One good point of bzImage is that when kernel/head.S is called, it doesn't notice the extra work, and everything goes forward as usual."

Rubini, 4.

1.4 accessing and decompressing said at least a portion of said boot data in said compressed form from said memory; and	Rubini, as evidenced by the example citations below, discloses “accessing and decompressing said at least a portion of said boot data in said compressed form from said memory; and”

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing and decompressing said at least a portion of said boot data in said compressed form from said memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.

Rubini discloses this limitation:

> The firmware usually checks the hardware's functionality, retrieves part (or all) of the kernel from a storage medium and executes it. This first part of the kernel must load the rest of itself and initialize the whole system.”

Rubini, 1.

> “Booting zImage and bzImage
>
> Even though it's rare to boot the system without a boot loader, it is still possible to do so by copying the raw kernel to a floppy disk. The command cat zImage >/dev/fd0 works perfectly on Linux, although some other Unix systems can do the task reliably only by using the dd command. Without going into detail, the raw floppy image created by zImage can then be configured using the rdev program.
>
> The file called zImage is the compressed kernel image that resides in arch/i386/boot after either make zImage or make boot is executed—the latter invocation is the one I prefer, as it works unchanged on other platforms. If you built a “big zImage” instead, the kernel file created is called bzImage and resides in the same directory.
>
> Booting an x86 kernel is a tricky task because of the limited amount of available memory. The Linux kernel tries to maximize usage of the low 640 kilobytes by moving itself around several times. Let's look at the steps performed by a zImage kernel in detail; all of the following path names are relative to the arch/i386/boot directory.

- The first sector (executing at 0x7c00) moves itself to 0x90000 and loads subsequent sectors after itself, getting them from the boot device using the firmware's functions to access the disk. The rest of the kernel is then loaded to address 0x10000, allowing for a maximum size of half a megabyte of data—remember, this is the compressed image. The boot sector code lives in bootsect.S, a real-mode assembly file.

- Then code at 0x90200 (defined in setup.S) takes care of some hardware initialization and allows the default text mode (video.S) to be changed. Text mode selection is a compile-time option from 2.1.9 onwards.
- Later, all the kernel is moved from 0x10000 (64K) to 0x1000 (4K). This move overwrites BIOS data stored in RAM, so BIOS calls can no longer be performed. The first physical page is not touched because it is the so-called "zero-page", used in handling virtual memory.
- At this point, setup.S enters protected mode and jumps to 0x1000, where the kernel lives. All the available memory can be accessed now, and the system can begin to run."

Rubini, 2-3.

"The steps just described were once the whole story of booting when the kernel was small enough to fit in half a megabyte of memory—the address range between 0x10000 and 0x90000. As features were added to the system, the kernel became larger than half a megabyte and could no longer be moved to 0x1000. Thus, code at 0x1000 is no longer th Linux kernel, instead the "gunzip" part of the gzip program resides at that address. The following additional steps are now needed t uncompress the kernel and execute it:

- head.S in the compressed directory is at 0x1000, and is in charge of "gunzipping" the kernel; it calls the function decompress_kernel, defined in compressed/misc.c, which in turns calls inflate which writes its output starting at address 0x100000 (1MB). High memory can now be accessed, because the processor is definitely out of its limited boot environment—the "real" mode.
- After decompression, head.S jumps to the actual beginning of the kernel. The relevant code is in ../kernel/head.S, outside of the boot directory.

The boot process is now over, and head.S (i.e., the code found at 0x100000 that used to be at 0x1000 before introducing compressed boots) can complete processor initialization and call start_kernel(). Code for all functions after this step is written in C."

Rubini, 3.

"The boot steps shown above rely on the assumption that the compressed kernel can fit in half a megabyte of space. While this is true most of the time, a system stuffed with device drivers might not fit into this space. For example, kernels used in installation disks can easily outgrow the available space. Some new method is needed to solve the problem—this new method is called bzImage and was introduced in kernel version 1.3.73.

A bzImage is generated by issuing make bzImage from the top level Linux source directory. This kind of kernel image boots similarly to zImage, with a few changes:

- When the system is loaded to address 0x10000, a little helper routine is called after loading each 64K data block. The helper routine moves the data block to high memory by using a special BIOS call. Only the newer BIOS versions implement this functionality, and so, make boot still builds the conventional zImage, though this may change in the near future.
- setup.S doesn't move the system back to 0x1000 (4K) but, after entering protected mode, jumps instead directly to address 0x100000 (1MB) where data has been moved by the BIOS in the previous step."

Rubini, 3-4.

"The rule for building the big compressed image can be read from Makefile; it affects several files in arch/i386/boot. One good point of bzImage is that when kernel/head.S is called, it doesn't notice the extra work, and everything goes forward as usual."

Rubini, 4.

"The decompresser found at 1MB writes the uncompressed kernel image into low memory until it is exhausted, and then into high memory after the compressed image. The two pieces are then reassembled to the address 0x100000 (1MB). Several memory moves are needed to perform the task correctly."

Rubini, 4.

1.5 utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine.	Rubini, as evidenced by the example citations below, discloses "utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rubini discloses this limitation: The firmware usually checks the hardware's functionality, retrieves part (or all) of the kernel from a storage medium and executes it. This first part of the kernel must load the rest of itself and initialize the whole system." Rubini, 1. "Booting zImage and bzImage Even though it's rare to boot the system without a boot loader, it is still possible to do so by copying the raw kernel to a floppy disk. The command cat zImage >/dev/fd0 works perfectly on Linux, although some other Unix systems can do the task reliably only by using the dd command. Without going into detail, the raw floppy image created by zImage can then be configured using the rdev program. The file called zImage is the compressed kernel image that resides in arch/i386/boot after either make zImage or make boot is executed—the latter invocation is the one I prefer, as it works unchanged on other platforms. If you built a "big zImage" instead, the kernel file created is called bzImage and resides in the same directory. Booting an x86 kernel is a tricky task because of the limited amount of available memory. The Linux kernel tries to maximize usage of the low 640 kilobytes by moving itself around several times. Let's look at the steps performed by a zImage kernel in detail; all of the following path	

names are relative to the arch/i386/boot directory.

- The first sector (executing at 0x7c00) moves itself to 0x90000 and loads subsequent sectors after itself, getting them from the boot device using the firmware's functions to access the disk. The rest of the kernel is then loaded to address 0x10000, allowing for a maximum size of half a megabyte of data—remember, this is the compressed image. The boot sector code lives in bootsect.S, a real-mode assembly file.
- Then code at 0x90200 (defined in setup.S) takes care of some hardware initialization and allows the default text mode (video.S) to be changed. Text mode selection is a compile-time option from 2.1.9 onwards.
- Later, all the kernel is moved from 0x10000 (64K) to 0x1000 (4K). This move overwrites BIOS data stored in RAM, so BIOS calls can no longer be performed. The first physical page is not touched because it is the so-called "zero-page", used in handling virtual memory.
- At this point, setup.S enters protected mode and jumps to 0x1000, where the kernel lives. All the available memory can be accessed now, and the system can begin to run."

Rubini, 2-3.

"The steps just described were once the whole story of booting when the kernel was small enough to fit in half a megabyte of memory—the address range between 0x10000 and 0x90000. As features were added to the system, the kernel became larger than half a megabyte and could no longer be moved to 0x1000. Thus, code at 0x1000 is no longer th Linux kernel, instead the "gunzip" part of the gzip program resides at that address. The following additional steps are now needed t uncompress the kernel and execute it:

- head.S in the compressed directory is at 0x1000, and is in charge of "gunzipping" the kernel; it calls the function decompress_kernel, defined in compressed/misc.c, which in turns calls inflate which writes its output starting at address 0x100000 (1MB). High memory can now be accessed, because the processor is definitely out of its limited boot environment—the "real" mode.
- After decompression, head.S jumps to the actual beginning of the kernel. The relevant code is in ../kernel/head.S, outside of the boot directory.

The boot process is now over, and head.S (i.e., the code found at 0x100000 that used to be at 0x1000 before introducing compressed boots) can complete processor initialization and call start_kernel(). Code for all functions after this step is written in C."

Rubini, 3.

> "The decompresser found at 1MB writes the uncompressed kernel image into low memory until it is exhausted, and then into high memory after the compressed image. The two pieces are then reassembled to the address 0x100000 (1MB). Several memory moves are needed to perform the task correctly."

Rubini, 4.

2. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system.	Rubini, as evidenced by the example citations below, discloses "said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rubini discloses this limitation: *See* Claims 1.1, 1.4, and 1.5 above.	

3. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system.	Rubini, as evidenced by the example citations below, discloses "said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rubini discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

4. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an application program and an operating system of said computer system.	Rubini, as evidenced by the example citations below, discloses "said decompressed at least a portion of said boot data comprises program code associated with an application program and an operating system of said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rubini discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

Appendix B36
Invalidity of U.S. Patent 8,090,936 based on Rubini

5. The method of claim 1, wherein said preloading is performed by a data storage controller connected to said boot device.	Rubini, as evidenced by the example citations below, discloses "said preloading is performed by a data storage controller connected to said boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said preloading is performed by a data storage controller connected to said boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rubini discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

6. The method of claim 1, further comprising updating the list of boot data.	Rubini, as evidenced by the example citations below, discloses "updating the list of boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Rubini discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also

> "Later, all the kernel is moved from 0x10000 (64K) to 0x1000 (4K). This move overwrites BIOS data stored in RAM, so BIOS calls can no longer be performed. The first physical page is not touched because it is the so-called "zero-page", used in handling virtual memory."

Rubini, 2.

8. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form.	Rubini, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rubini discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

9. The method of claim 1, wherein a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form.	Rubini, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rubini discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

11.1. a processor;	Rubini, as evidenced by the example citations below, discloses “a processor.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Rubini discloses this limitation: *See* Claim 1.2 above.	

11.2. a memory; and	Rubini, as evidenced by the example citations below, discloses “a memory.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Rubini discloses this limitation: *See* Claims 1.3, and 1.4 above.	

11.3.1 a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device,	Rubini, as evidenced by the example citations below, discloses “a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Rubini discloses this limitation: *See* Claims 1.1, 1.3, 1.4 above.	

11.3.2 said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system; and,	Rubini, as evidenced by the example citations below, discloses "said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rubini discloses this limitation: *See* Claims 1.3, 1.4, and 1.5 above.	

11.4 a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data.	Rubini, as evidenced by the example citations below, discloses "a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rubini discloses this limitation: *See* Claims 1.1 and 1.5 above.	

12. The system of claim 11, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program.	Rubini, as evidenced by the example citations below, discloses "said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rubini discloses this limitation: *See* Claims 1.1, 3, and 11.3.1 above.	

13. The system of claim 11, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system.	Rubini, as evidenced by the example citations below, discloses "wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rubini discloses this limitation: *See* Claims 1.1, 1.3, 3, 5 and 11.3.1 and 11.3.2 above.	

15. The system of claim 11, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form.	Rubini, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rubini discloses this limitation: *See* Claims 1, 8, and 11 above.	

16. The system of claim 11, wherein a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form.	Rubini, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Rubini discloses this limitation: *See* Claims 1, 9, and 11 above.	

Appendix B37
Invalidity of U.S. Patent 8,090,936 based on Wynn

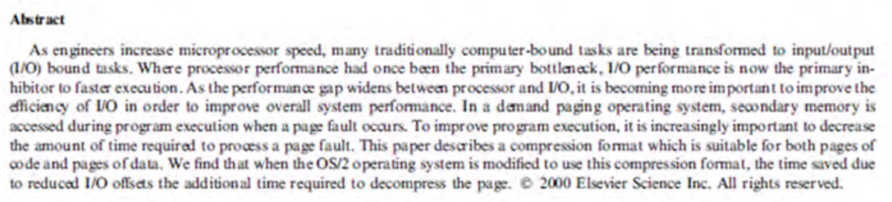
Wynn, et al. "The effect of compression on performance in a demand paging operating system," The Journal of Systems and Software (2000) ("Wynn Article") and Wynn, "The Effect of Compression on Performance in a Demand Paging Operating System," 1997 ("Wynn Thesis") (collectively, "Wynn"), alone or in combination, invalidate claims 1-6, 8-9, 11-13, and 15-16 of United States Patent No. 8,090,936 ("the '936 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '936 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

1.1 maintaining a list of boot data used for booting a computer system, wherein at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device;	Wynn, as evidenced by the exemplary citations below, discloses "maintaining a list of boot data used for booting a computer system, wherein at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system, wherein at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

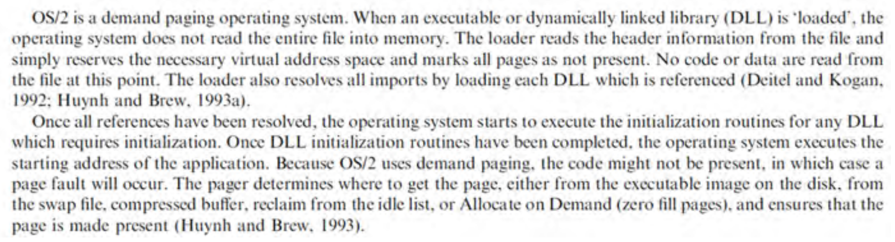
Wynn discloses this limitation:



Abstract

As engineers increase microprocessor speed, many traditionally computer-bound tasks are being transformed to input/output (I/O) bound tasks. Where processor performance had once been the primary bottleneck, I/O performance is now the primary inhibitor to faster execution. As the performance gap widens between processor and I/O, it is becoming more important to improve the efficiency of I/O in order to improve overall system performance. In a demand paging operating system, secondary memory is accessed during program execution when a page fault occurs. To improve program execution, it is increasingly important to decrease the amount of time required to process a page fault. This paper describes a compression format which is suitable for both pages of code and pages of data. We find that when the OS/2 operating system is modified to use this compression format, the time saved due to reduced I/O offsets the additional time required to decompress the page. © 2000 Elsevier Science Inc. All rights reserved.

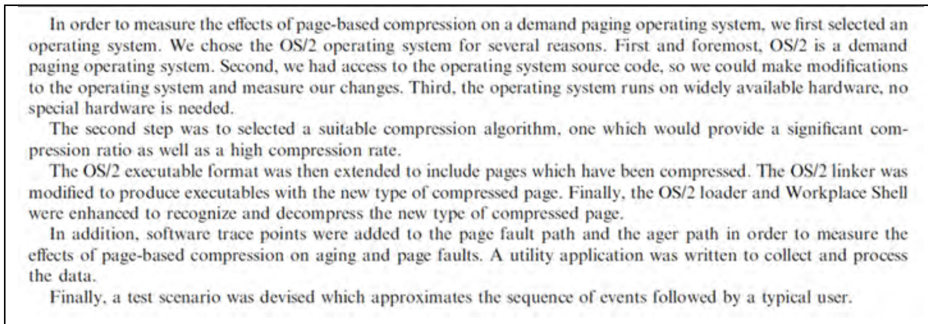
Wynn Article, Abstract



OS/2 is a demand paging operating system. When an executable or dynamically linked library (DLL) is 'loaded', the operating system does not read the entire file into memory. The loader reads the header information from the file and simply reserves the necessary virtual address space and marks all pages as not present. No code or data are read from the file at this point. The loader also resolves all imports by loading each DLL which is referenced (Deitel and Kogan, 1992; Huynh and Brew, 1993a).

Once all references have been resolved, the operating system starts to execute the initialization routines for any DLL which requires initialization. Once DLL initialization routines have been completed, the operating system executes the starting address of the application. Because OS/2 uses demand paging, the code might not be present, in which case a page fault will occur. The pager determines where to get the page, either from the executable image on the disk, from the swap file, compressed buffer, reclaim from the idle list, or Allocate on Demand (zero fill pages), and ensures that the page is made present (Huynh and Brew, 1993).

Wynn Article, 2.2



In order to measure the effects of page-based compression on a demand paging operating system, we first selected an operating system. We chose the OS/2 operating system for several reasons. First and foremost, OS/2 is a demand paging operating system. Second, we had access to the operating system source code, so we could make modifications to the operating system and measure our changes. Third, the operating system runs on widely available hardware, no special hardware is needed.

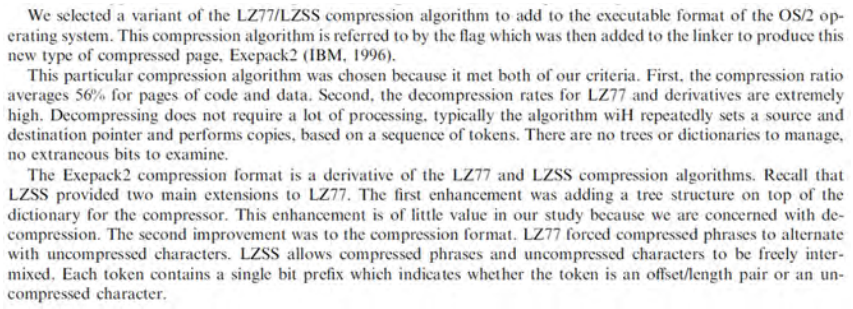
The second step was to selected a suitable compression algorithm, one which would provide a significant compression ratio as well as a high compression rate.

The OS/2 executable format was then extended to include pages which have been compressed. The OS/2 linker was modified to produce executables with the new type of compressed page. Finally, the OS/2 loader and Workplace Shell were enhanced to recognize and decompress the new type of compressed page.

In addition, software trace points were added to the page fault path and the ager path in order to measure the effects of page-based compression on aging and page faults. A utility application was written to collect and process the data.

Finally, a test scenario was devised which approximates the sequence of events followed by a typical user.

Wynn Article, 3.1

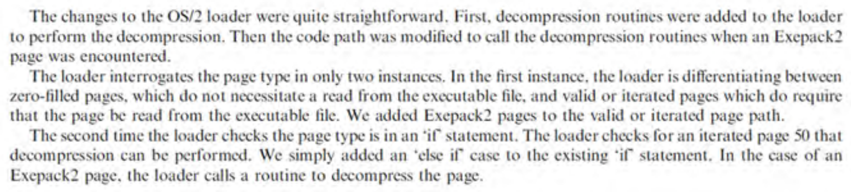


We selected a variant of the LZ77/LZSS compression algorithm to add to the executable format of the OS/2 operating system. This compression algorithm is referred to by the flag which was then added to the linker to produce this new type of compressed page, Exepack2 (IBM, 1996).

This particular compression algorithm was chosen because it met both of our criteria. First, the compression ratio averages 56% for pages of code and data. Second, the decompression rates for LZ77 and derivatives are extremely high. Decompressing does not require a lot of processing, typically the algorithm wiH repeatedly sets a source and destination pointer and performs copies, based on a sequence of tokens. There are no trees or dictionaries to manage, no extraneous bits to examine.

The Exepack2 compression format is a derivative of the LZ77 and LZSS compression algorithms. Recall that LZSS provided two main extensions to LZ77. The first enhancement was adding a tree structure on top of the dictionary for the compressor. This enhancement is of little value in our study because we are concerned with decompression. The second improvement was to the compression format. LZ77 forced compressed phrases to alternate with uncompressed characters. LZSS allows compressed phrases and uncompressed characters to be freely intermixed. Each token contains a single bit prefix which indicates whether the token is an offset/length pair or an uncompressed character.

Wynn Article, 3.2

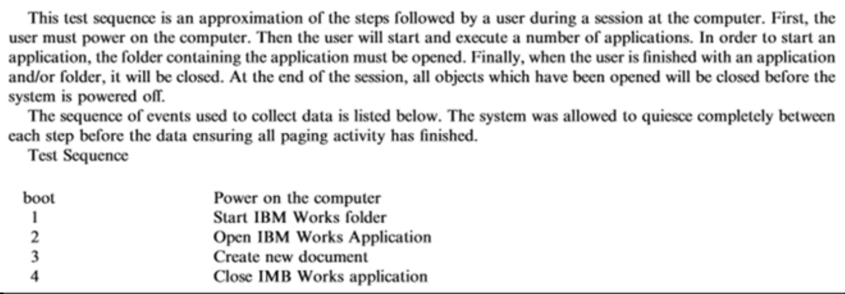


The changes to the OS/2 loader were quite straightforward. First, decompression routines were added to the loader to perform the decompression. Then the code path was modified to call the decompression routines when an Exepack2 page was encountered.

The loader interrogates the page type in only two instances. In the first instance, the loader is differentiating between zero-filled pages, which do not necessitate a read from the executable file, and valid or iterated pages which do require that the page be read from the executable file. We added Exepack2 pages to the valid or iterated page path.

The second time the loader checks the page type is in an 'if' statement. The loader checks for an iterated page 50 that decompression can be performed. We simply added an 'else if' case to the existing 'if' statement. In the case of an Exepack2 page, the loader calls a routine to decompress the page.

Wynn Article, 3.4

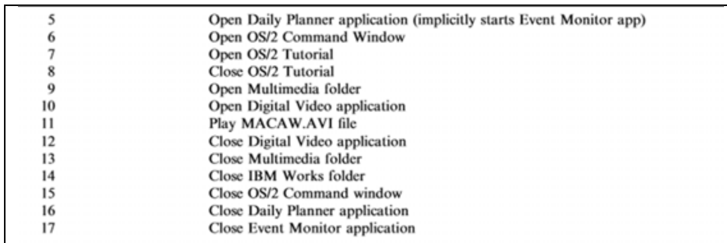


This test sequence is an approximation of the steps followed by a user during a session at the computer. First, the user must power on the computer. Then the user will start and execute a number of applications. In order to start an application, the folder containing the application must be opened. Finally, when the user is finished with an application and/or folder, it will be closed. At the end of the session, all objects which have been opened will be closed before the system is powered off.

The sequence of events used to collect data is listed below. The system was allowed to quiesce completely between each step before the data ensuring all paging activity has finished.

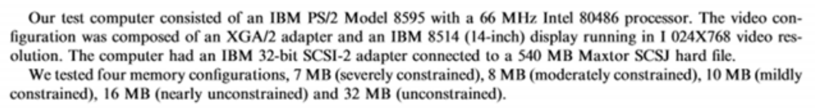
Test Sequence

boot	**Power on the computer**
1	**Start IBM Works folder**
2	**Open IBM Works Application**
3	**Create new document**
4	**Close IMB Works application**



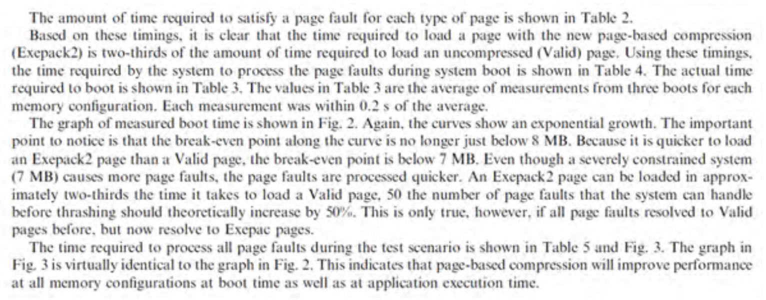
5	Open Daily Planner application (implicitly starts Event Monitor app)
6	Open OS/2 Command Window
7	Open OS/2 Tutorial
8	Close OS/2 Tutorial
9	Open Multimedia folder
10	Open Digital Video application
11	Play MACAW.AVI file
12	Close Digital Video application
13	Close Multimedia folder
14	Close IBM Works folder
15	Close OS/2 Command window
16	Close Daily Planner application
17	Close Event Monitor application

Wynn Article, 3.8



Our test computer consisted of an IBM PS/2 Model 8595 with a 66 MHz Intel 80486 processor. The video configuration was composed of an XGA/2 adapter and an IBM 8514 (14-inch) display running in I 024X768 video resolution. The computer had an IBM 32-bit SCSI-2 adapter connected to a 540 MB Maxtor SCSJ hard file.
We tested four memory configurations, 7 MB (severely constrained), 8 MB (moderately constrained), 10 MB (mildly constrained), 16 MB (nearly unconstrained) and 32 MB (unconstrained).

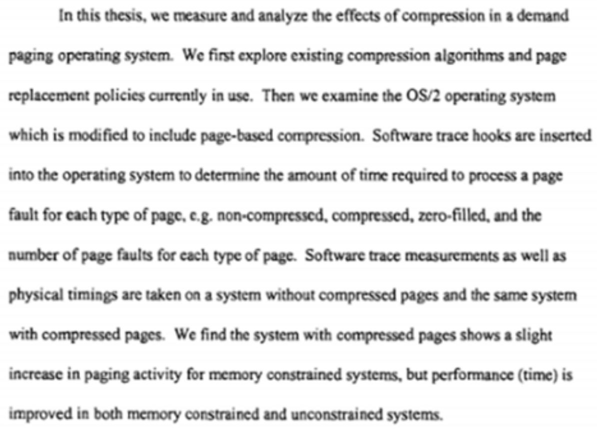
Wynn Article, 3.9



The amount of time required to satisfy a page fault for each type of page is shown in Table 2.
Based on these timings, it is clear that the time required to load a page with the new page-based compression (Exepack2) is two-thirds of the amount of time required to load an uncompressed (Valid) page. Using these timings, the time required by the system to process the page faults during system boot is shown in Table 4. The actual time required to boot is shown in Table 3. The values in Table 3 are the average of measurements from three boots for each memory configuration. Each measurement was within 0.2 s of the average.
The graph of measured boot time is shown in Fig. 2. Again, the curves show an exponential growth. The important point to notice is that the break-even point along the curve is no longer just below 8 MB. Because it is quicker to load an Exepack2 page than a Valid page, the break-even point is below 7 MB. Even though a severely constrained system (7 MB) causes more page faults, the page faults are processed quicker. An Exepack2 page can be loaded in approximately two-thirds the time it takes to load a Valid page, 50 the number of page faults that the system can handle before thrashing should theoretically increase by 50%. This is only true, however, if all page faults resolved to Valid pages before, but now resolve to Exepac pages.
The time required to process all page faults during the test scenario is shown in Table 5 and Fig. 3. The graph in Fig. 3 is virtually identical to the graph in Fig. 2. This indicates that page-based compression will improve performance at all memory configurations at boot time as well as at application execution time.

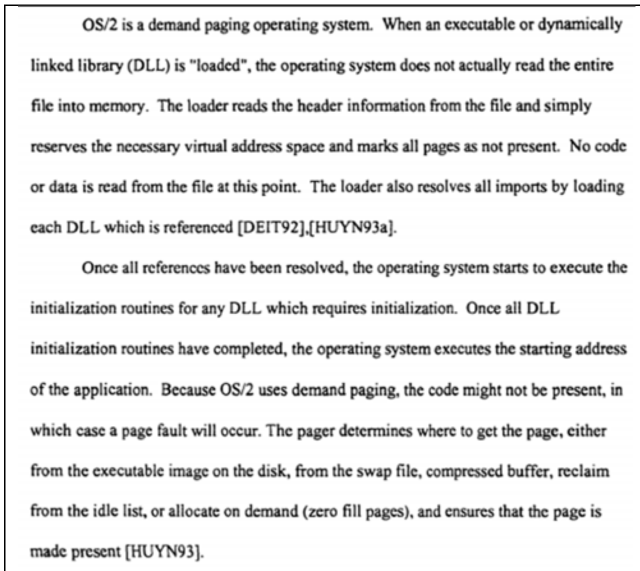
Wynn Article, 4.3

See also, Wynn Article, Table 3 and Fig. 2



In this thesis, we measure and analyze the effects of compression in a demand paging operating system. We first explore existing compression algorithms and page replacement policies currently in use. Then we examine the OS/2 operating system which is modified to include page-based compression. Software trace hooks are inserted into the operating system to determine the amount of time required to process a page fault for each type of page, e.g. non-compressed, compressed, zero-filled, and the number of page faults for each type of page. Software trace measurements as well as physical timings are taken on a system without compressed pages and the same system with compressed pages. We find the system with compressed pages shows a slight increase in paging activity for memory constrained systems, but performance (time) is improved in both memory constrained and unconstrained systems.

Wynn Thesis, Abstract



OS/2 is a demand paging operating system. When an executable or dynamically linked library (DLL) is "loaded", the operating system does not actually read the entire file into memory. The loader reads the header information from the file and simply reserves the necessary virtual address space and marks all pages as not present. No code or data is read from the file at this point. The loader also resolves all imports by loading each DLL which is referenced [DEIT92],[HUYN93a].

Once all references have been resolved, the operating system starts to execute the initialization routines for any DLL which requires initialization. Once all DLL initialization routines have completed, the operating system executes the starting address of the application. Because OS/2 uses demand paging, the code might not be present, in which case a page fault will occur. The pager determines where to get the page, either from the executable image on the disk, from the swap file, compressed buffer, reclaim from the idle list, or allocate on demand (zero fill pages), and ensures that the page is made present [HUYN93].

Wynn Thesis, 4.2

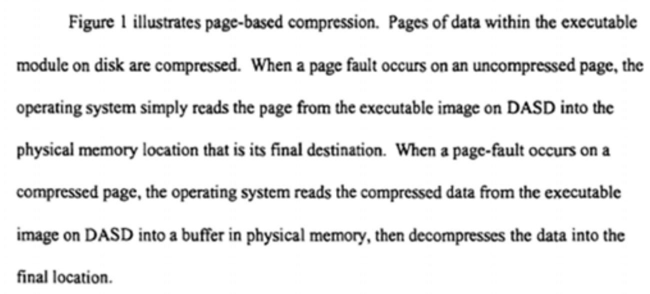
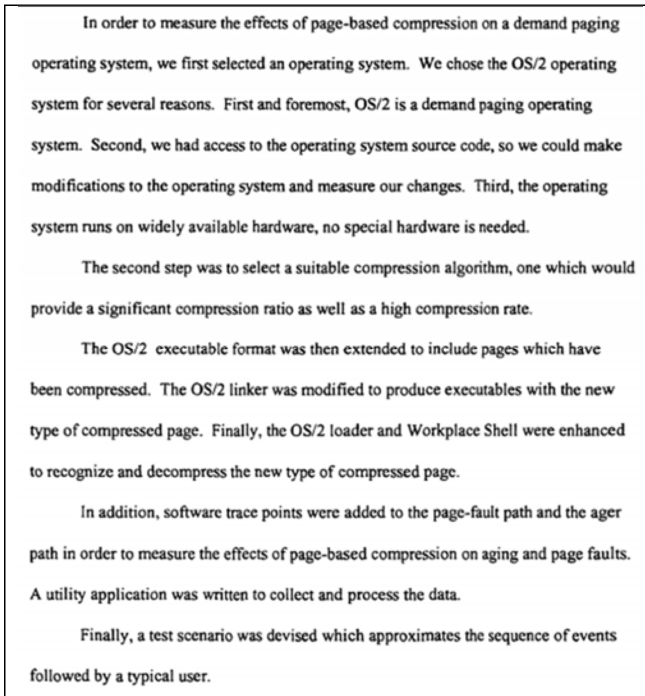


Figure 1 illustrates page-based compression. Pages of data within the executable module on disk are compressed. When a page fault occurs on an uncompressed page, the operating system simply reads the page from the executable image on DASD into the physical memory location that is its final destination. When a page-fault occurs on a compressed page, the operating system reads the compressed data from the executable image on DASD into a buffer in physical memory, then decompresses the data into the final location.



In order to measure the effects of page-based compression on a demand paging operating system, we first selected an operating system. We chose the OS/2 operating system for several reasons. First and foremost, OS/2 is a demand paging operating system. Second, we had access to the operating system source code, so we could make modifications to the operating system and measure our changes. Third, the operating system runs on widely available hardware, no special hardware is needed.

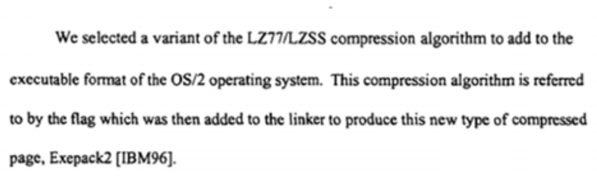
The second step was to select a suitable compression algorithm, one which would provide a significant compression ratio as well as a high compression rate.

The OS/2 executable format was then extended to include pages which have been compressed. The OS/2 linker was modified to produce executables with the new type of compressed page. Finally, the OS/2 loader and Workplace Shell were enhanced to recognize and decompress the new type of compressed page.

In addition, software trace points were added to the page-fault path and the ager path in order to measure the effects of page-based compression on aging and page faults. A utility application was written to collect and process the data.

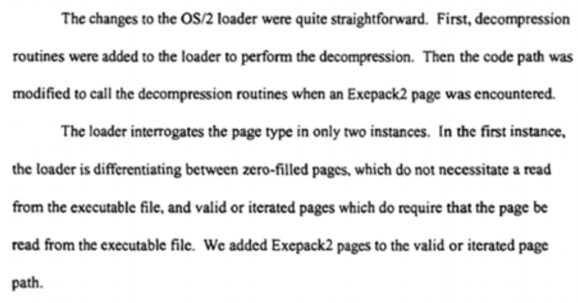
Finally, a test scenario was devised which approximates the sequence of events followed by a typical user.

Wynn Thesis, 5.1



We selected a variant of the LZ77/LZSS compression algorithm to add to the executable format of the OS/2 operating system. This compression algorithm is referred to by the flag which was then added to the linker to produce this new type of compressed page, Exepack2 [IBM96].

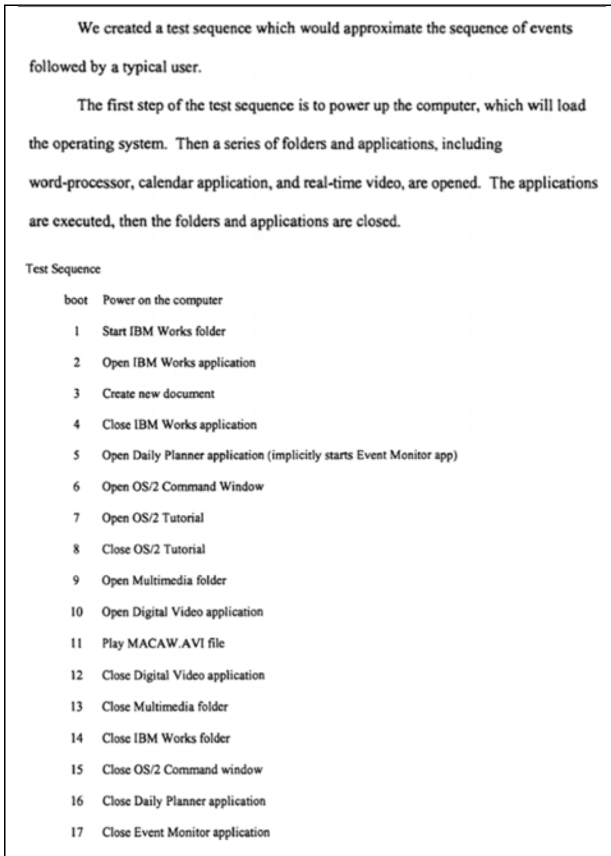
Wynn Thesis, 5.2



The changes to the OS/2 loader were quite straightforward. First, decompression routines were added to the loader to perform the decompression. Then the code path was modified to call the decompression routines when an Exepack2 page was encountered.

The loader interrogates the page type in only two instances. In the first instance, the loader is differentiating between zero-filled pages, which do not necessitate a read from the executable file, and valid or iterated pages which do require that the page be read from the executable file. We added Exepack2 pages to the valid or iterated page path.

Wynn Thesis, 5.5



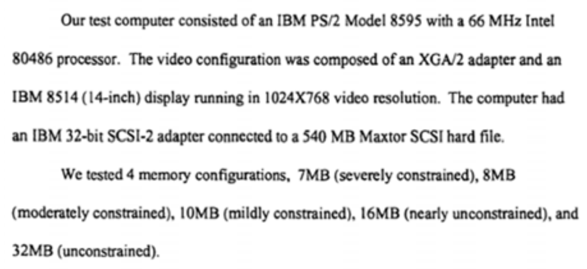
We created a test sequence which would approximate the sequence of events followed by a typical user.

The first step of the test sequence is to power up the computer, which will load the operating system. Then a series of folders and applications, including word-processor, calendar application, and real-time video, are opened. The applications are executed, then the folders and applications are closed.

Test Sequence

boot	Power on the computer
1	Start IBM Works folder
2	Open IBM Works application
3	Create new document
4	Close IBM Works application
5	Open Daily Planner application (implicitly starts Event Monitor app)
6	Open OS/2 Command Window
7	Open OS/2 Tutorial
8	Close OS/2 Tutorial
9	Open Multimedia folder
10	Open Digital Video application
11	Play MACAW.AVI file
12	Close Digital Video application
13	Close Multimedia folder
14	Close IBM Works folder
15	Close OS/2 Command window
16	Close Daily Planner application
17	Close Event Monitor application

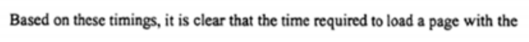
Wynn Thesis, 5.9



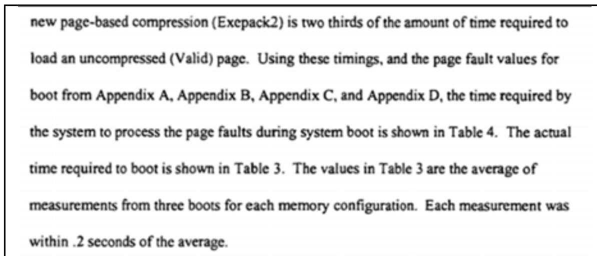
Our test computer consisted of an IBM PS/2 Model 8595 with a 66 MHz Intel 80486 processor. The video configuration was composed of an XGA/2 adapter and an IBM 8514 (14-inch) display running in 1024X768 video resolution. The computer had an IBM 32-bit SCSI-2 adapter connected to a 540 MB Maxtor SCSI hard file.

We tested 4 memory configurations, 7MB (severely constrained), 8MB (moderately constrained), 10MB (mildly constrained), 16MB (nearly unconstrained), and 32MB (unconstrained).

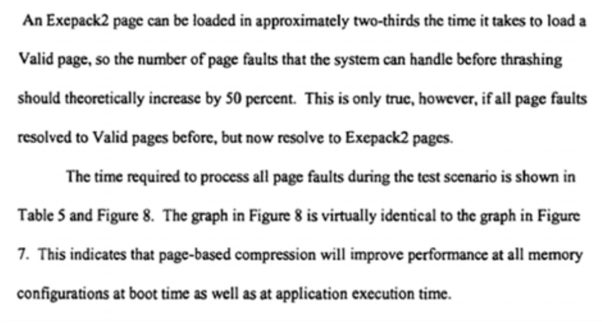
Wynn Thesis, 5.10



Based on these timings, it is clear that the time required to load a page with the

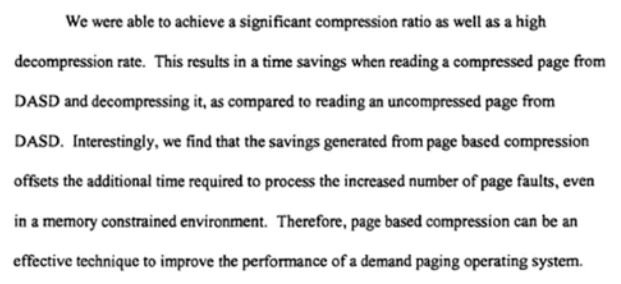

new page-based compression (Exepack2) is two thirds of the amount of time required to load an uncompressed (Valid) page. Using these timings, and the page fault values for boot from Appendix A, Appendix B, Appendix C, and Appendix D, the time required by the system to process the page faults during system boot is shown in Table 4. The actual time required to boot is shown in Table 3. The values in Table 3 are the average of measurements from three boots for each memory configuration. Each measurement was within .2 seconds of the average.

Wynn Thesis, 6.3


An Exepack2 page can be loaded in approximately two-thirds the time it takes to load a Valid page, so the number of page faults that the system can handle before thrashing should theoretically increase by 50 percent. This is only true, however, if all page faults resolved to Valid pages before, but now resolve to Exepack2 pages.

The time required to process all page faults during the test scenario is shown in Table 5 and Figure 8. The graph in Figure 8 is virtually identical to the graph in Figure 7. This indicates that page-based compression will improve performance at all memory configurations at boot time as well as at application execution time.

Wynn Thesis, 6.3


We were able to achieve a significant compression ratio as well as a high decompression rate. This results in a time savings when reading a compressed page from DASD and decompressing it, as compared to reading an uncompressed page from DASD. Interestingly, we find that the savings generated from page based compression offsets the additional time required to process the increased number of page faults, even in a memory constrained environment. Therefore, page based compression can be an effective technique to improve the performance of a demand paging operating system.

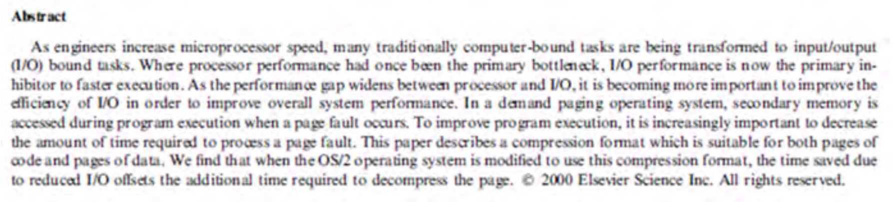
Wynn Thesis, 7.2

See also Wynn Thesis, Table 2-4, Fig. 7

1.2 initializing a central processing unit of said computer system;	Wynn, as evidenced by the example citations below, discloses "initializing a central processing unit of said computer system;"

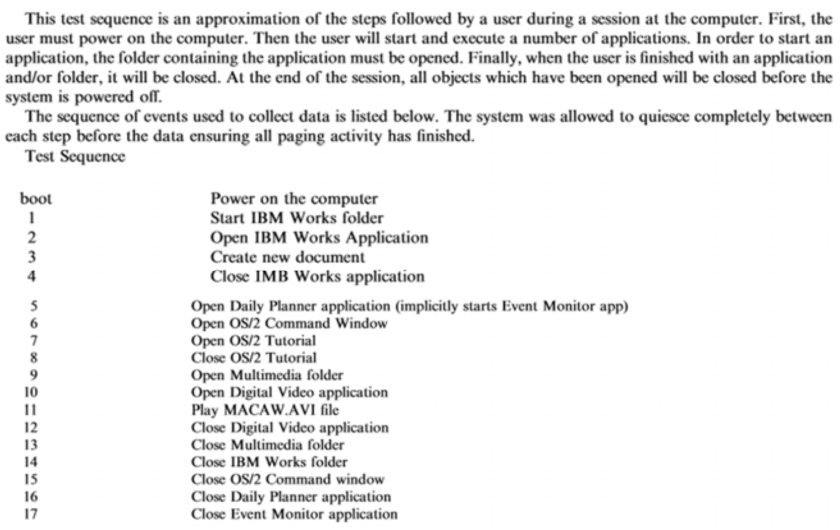
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Wynn discloses this limitation:


Abstract

As engineers increase microprocessor speed, many traditionally computer-bound tasks are being transformed to input/output (I/O) bound tasks. Where processor performance had once been the primary bottleneck, I/O performance is now the primary inhibitor to faster execution. As the performance gap widens between processor and I/O, it is becoming more important to improve the efficiency of I/O in order to improve overall system performance. In a demand paging operating system, secondary memory is accessed during program execution when a page fault occurs. To improve program execution, it is increasingly important to decrease the amount of time required to process a page fault. This paper describes a compression format which is suitable for both pages of code and pages of data. We find that when the OS/2 operating system is modified to use this compression format, the time saved due to reduced I/O offsets the additional time required to decompress the page. © 2000 Elsevier Science Inc. All rights reserved.

Wynn Article, Abstract

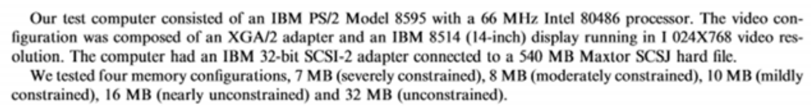

This test sequence is an approximation of the steps followed by a user during a session at the computer. First, the user must power on the computer. Then the user will start and execute a number of applications. In order to start an application, the folder containing the application must be opened. Finally, when the user is finished with an application and/or folder, it will be closed. At the end of the session, all objects which have been opened will be closed before the system is powered off.

The sequence of events used to collect data is listed below. The system was allowed to quiesce completely between each step before the data ensuring all paging activity has finished.

Test Sequence

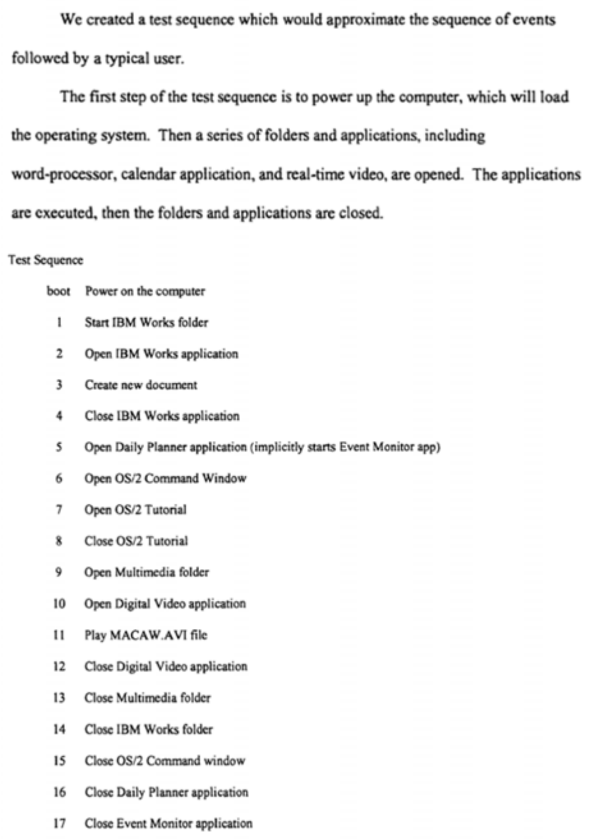
boot	Power on the computer
1	Start IBM Works folder
2	Open IBM Works Application
3	Create new document
4	Close IMB Works application
5	Open Daily Planner application (implicitly starts Event Monitor app)
6	Open OS/2 Command Window
7	Open OS/2 Tutorial
8	Close OS/2 Tutorial
9	Open Multimedia folder
10	Open Digital Video application
11	Play MACAW.AVI file
12	Close Digital Video application
13	Close Multimedia folder
14	Close IBM Works folder
15	Close OS/2 Command window
16	Close Daily Planner application
17	Close Event Monitor application

Wynn Article, 3.8


Our test computer consisted of an IBM PS/2 Model 8595 with a 66 MHz Intel 80486 processor. The video configuration was composed of an XGA/2 adapter and an IBM 8514 (14-inch) display running in 1 024X768 video resolution. The computer had an IBM 32-bit SCSI-2 adapter connected to a 540 MB Maxtor SCSJ hard file.

We tested four memory configurations, 7 MB (severely constrained), 8 MB (moderately constrained), 10 MB (mildly constrained), 16 MB (nearly unconstrained) and 32 MB (unconstrained).

Wynn Article, 3.9



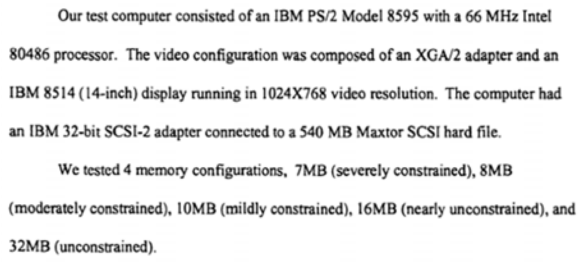
We created a test sequence which would approximate the sequence of events followed by a typical user.

The first step of the test sequence is to power up the computer, which will load the operating system. Then a series of folders and applications, including word-processor, calendar application, and real-time video, are opened. The applications are executed, then the folders and applications are closed.

Test Sequence

boot	Power on the computer
1	Start IBM Works folder
2	Open IBM Works application
3	Create new document
4	Close IBM Works application
5	Open Daily Planner application (implicitly starts Event Monitor app)
6	Open OS/2 Command Window
7	Open OS/2 Tutorial
8	Close OS/2 Tutorial
9	Open Multimedia folder
10	Open Digital Video application
11	Play MACAW.AVI file
12	Close Digital Video application
13	Close Multimedia folder
14	Close IBM Works folder
15	Close OS/2 Command window
16	Close Daily Planner application
17	Close Event Monitor application

Wynn Thesis, 5.9



Our test computer consisted of an IBM PS/2 Model 8595 with a 66 MHz Intel 80486 processor. The video configuration was composed of an XGA/2 adapter and an IBM 8514 (14-inch) display running in 1024X768 video resolution. The computer had an IBM 32-bit SCSI-2 adapter connected to a 540 MB Maxtor SCSI hard file.

We tested 4 memory configurations, 7MB (severely constrained), 8MB (moderately constrained), 10MB (mildly constrained), 16MB (nearly unconstrained), and 32MB (unconstrained).

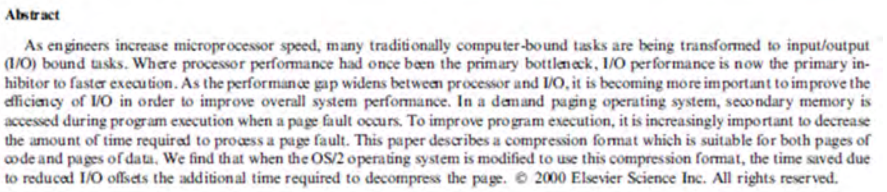
Wynn Thesis, 5.10

Appendix B37
Invalidity of U.S. Patent 8,090,936 based on Wynn

1.3 preloading said at least a portion of said boot data in compressed form from said boot device to a memory;	Wynn, as evidenced by the example citations below, discloses "preloading said at least a portion of said boot data in compressed form from said boot device to a memory;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading said at least a portion of said boot data in compressed form from said boot device to a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

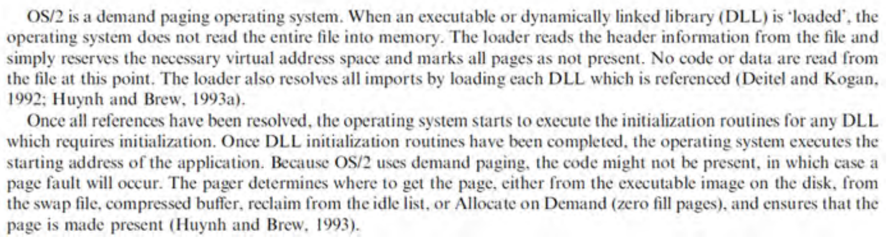
Wynn discloses this limitation:



Abstract

As engineers increase microprocessor speed, many traditionally computer-bound tasks are being transformed to input/output (I/O) bound tasks. Where processor performance had once been the primary bottleneck, I/O performance is now the primary inhibitor to faster execution. As the performance gap widens between processor and I/O, it is becoming more important to improve the efficiency of I/O in order to improve overall system performance. In a demand paging operating system, secondary memory is accessed during program execution when a page fault occurs. To improve program execution, it is increasingly important to decrease the amount of time required to process a page fault. This paper describes a compression format which is suitable for both pages of code and pages of data. We find that when the OS/2 operating system is modified to use this compression format, the time saved due to reduced I/O offsets the additional time required to decompress the page. © 2000 Elsevier Science Inc. All rights reserved.

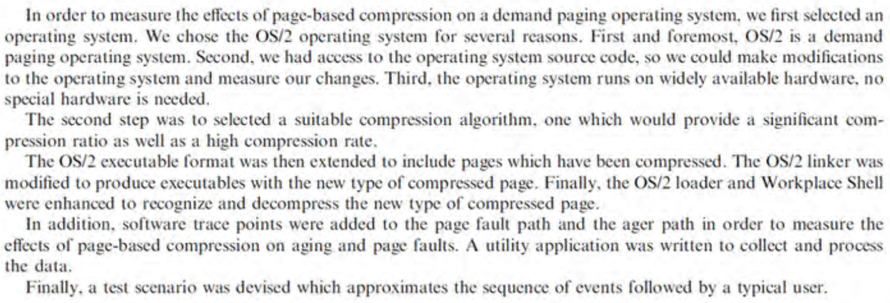
Wynn Article, Abstract



OS/2 is a demand paging operating system. When an executable or dynamically linked library (DLL) is 'loaded', the operating system does not read the entire file into memory. The loader reads the header information from the file and simply reserves the necessary virtual address space and marks all pages as not present. No code or data are read from the file at this point. The loader also resolves all imports by loading each DLL which is referenced (Deitel and Kogan, 1992; Huynh and Brew, 1993a).

Once all references have been resolved, the operating system starts to execute the initialization routines for any DLL which requires initialization. Once DLL initialization routines have been completed, the operating system executes the starting address of the application. Because OS/2 uses demand paging, the code might not be present, in which case a page fault will occur. The pager determines where to get the page, either from the executable image on the disk, from the swap file, compressed buffer, reclaim from the idle list, or Allocate on Demand (zero fill pages), and ensures that the page is made present (Huynh and Brew, 1993).

Wynn Article, 2.2



In order to measure the effects of page-based compression on a demand paging operating system, we first selected an operating system. We chose the OS/2 operating system for several reasons. First and foremost, OS/2 is a demand paging operating system. Second, we had access to the operating system source code, so we could make modifications to the operating system and measure our changes. Third, the operating system runs on widely available hardware, no special hardware is needed.

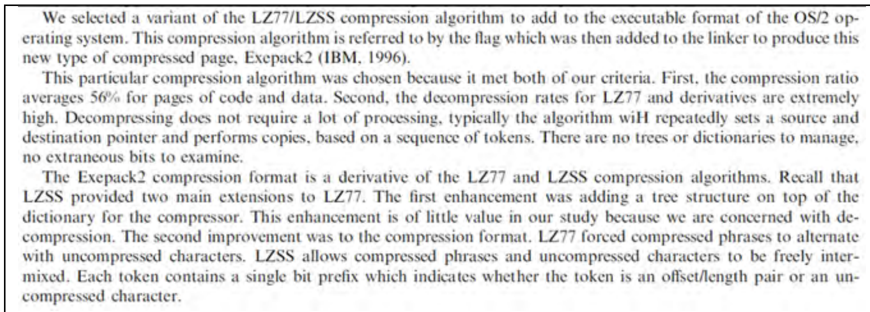
The second step was to selected a suitable compression algorithm, one which would provide a significant compression ratio as well as a high compression rate.

The OS/2 executable format was then extended to include pages which have been compressed. The OS/2 linker was modified to produce executables with the new type of compressed page. Finally, the OS/2 loader and Workplace Shell were enhanced to recognize and decompress the new type of compressed page.

In addition, software trace points were added to the page fault path and the ager path in order to measure the effects of page-based compression on aging and page faults. A utility application was written to collect and process the data.

Finally, a test scenario was devised which approximates the sequence of events followed by a typical user.

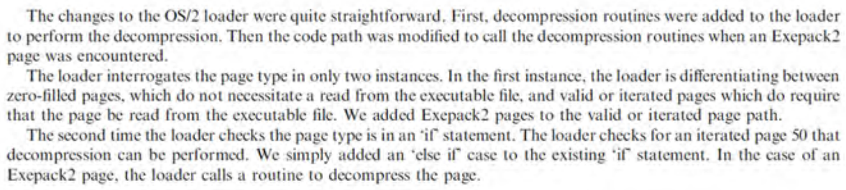
Wynn Article, 3.1


We selected a variant of the LZ77/LZSS compression algorithm to add to the executable format of the OS/2 operating system. This compression algorithm is referred to by the flag which was then added to the linker to produce this new type of compressed page, Exepack2 (IBM, 1996).

This particular compression algorithm was chosen because it met both of our criteria. First, the compression ratio averages 56% for pages of code and data. Second, the decompression rates for LZ77 and derivatives are extremely high. Decompressing does not require a lot of processing, typically the algorithm wiH repeatedly sets a source and destination pointer and performs copies, based on a sequence of tokens. There are no trees or dictionaries to manage, no extraneous bits to examine.

The Exepack2 compression format is a derivative of the LZ77 and LZSS compression algorithms. Recall that LZSS provided two main extensions to LZ77. The first enhancement was adding a tree structure on top of the dictionary for the compressor. This enhancement is of little value in our study because we are concerned with decompression. The second improvement was to the compression format. LZ77 forced compressed phrases to alternate with uncompressed characters. LZSS allows compressed phrases and uncompressed characters to be freely intermixed. Each token contains a single bit prefix which indicates whether the token is an offset/length pair or an uncompressed character.

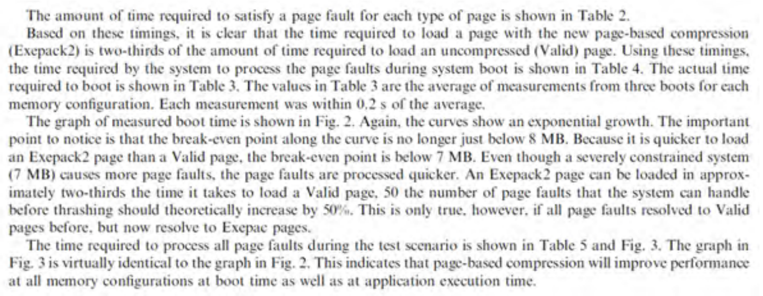
Wynn Article, 3.2


The changes to the OS/2 loader were quite straightforward. First, decompression routines were added to the loader to perform the decompression. Then the code path was modified to call the decompression routines when an Exepack2 page was encountered.

The loader interrogates the page type in only two instances. In the first instance, the loader is differentiating between zero-filled pages, which do not necessitate a read from the executable file, and valid or iterated pages which do require that the page be read from the executable file. We added Exepack2 pages to the valid or iterated page path.

The second time the loader checks the page type is in an 'if' statement. The loader checks for an iterated page 50 that decompression can be performed. We simply added an 'else if' case to the existing 'if' statement. In the case of an Exepack2 page, the loader calls a routine to decompress the page.

Wynn Article, 3.4


The amount of time required to satisfy a page fault for each type of page is shown in Table 2.

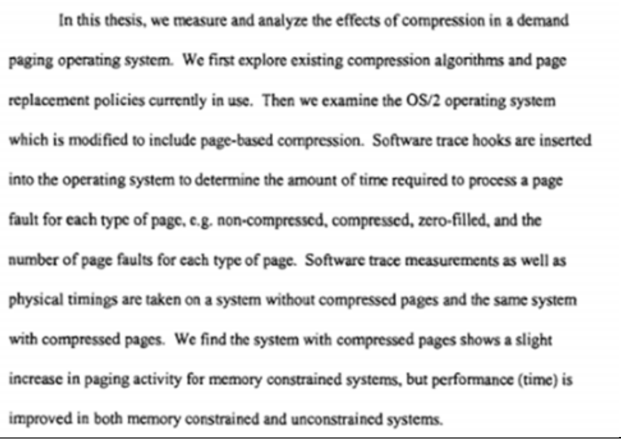
Based on these timings, it is clear that the time required to load a page with the new page-based compression (Exepack2) is two-thirds of the amount of time required to load an uncompressed (Valid) page. Using these timings, the time required by the system to process the page faults during system boot is shown in Table 4. The actual time required to boot is shown in Table 3. The values in Table 3 are the average of measurements from three boots for each memory configuration. Each measurement was within 0.2 s of the average.

The graph of measured boot time is shown in Fig. 2. Again, the curves show an exponential growth. The important point to notice is that the break-even point along the curve is no longer just below 8 MB. Because it is quicker to load an Exepack2 page than a Valid page, the break-even point is below 7 MB. Even though a severely constrained system (7 MB) causes more page faults, the page faults are processed quicker. An Exepack2 page can be loaded in approximately two-thirds the time it takes to load a Valid page, 50 the number of page faults that the system can handle before thrashing should theoretically increase by 50%. This is only true, however, if all page faults resolved to Valid pages before, but now resolve to Exepac pages.

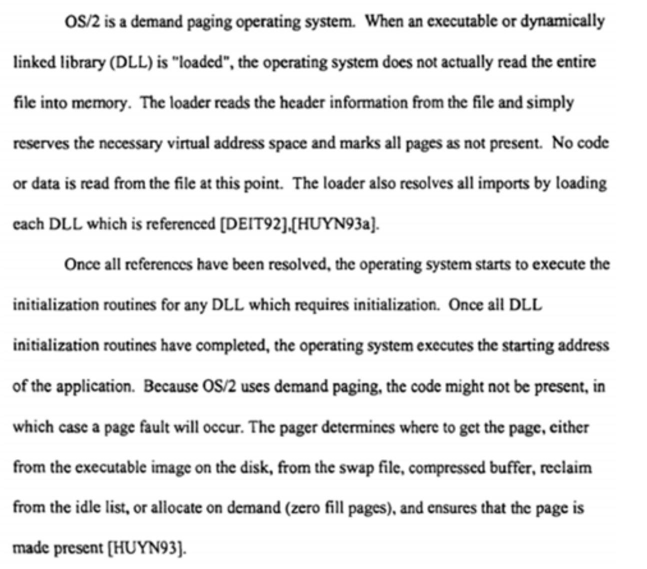
The time required to process all page faults during the test scenario is shown in Table 5 and Fig. 3. The graph in Fig. 3 is virtually identical to the graph in Fig. 2. This indicates that page-based compression will improve performance at all memory configurations at boot time as well as at application execution time.

Wynn Article, 4.3

See also, Wynn Article, Table 3 and Fig. 2


In this thesis, we measure and analyze the effects of compression in a demand paging operating system. We first explore existing compression algorithms and page replacement policies currently in use. Then we examine the OS/2 operating system which is modified to include page-based compression. Software trace hooks are inserted into the operating system to determine the amount of time required to process a page fault for each type of page, e.g. non-compressed, compressed, zero-filled, and the number of page faults for each type of page. Software trace measurements as well as physical timings are taken on a system without compressed pages and the same system with compressed pages. We find the system with compressed pages shows a slight increase in paging activity for memory constrained systems, but performance (time) is improved in both memory constrained and unconstrained systems.

Wynn Thesis, Abstract



OS/2 is a demand paging operating system. When an executable or dynamically linked library (DLL) is "loaded", the operating system does not actually read the entire file into memory. The loader reads the header information from the file and simply reserves the necessary virtual address space and marks all pages as not present. No code or data is read from the file at this point. The loader also resolves all imports by loading each DLL which is referenced [DEIT92],[HUYN93a].

Once all references have been resolved, the operating system starts to execute the initialization routines for any DLL which requires initialization. Once all DLL initialization routines have completed, the operating system executes the starting address of the application. Because OS/2 uses demand paging, the code might not be present, in which case a page fault will occur. The pager determines where to get the page, either from the executable image on the disk, from the swap file, compressed buffer, reclaim from the idle list, or allocate on demand (zero fill pages), and ensures that the page is made present [HUYN93].

Wynn Thesis, 4.2

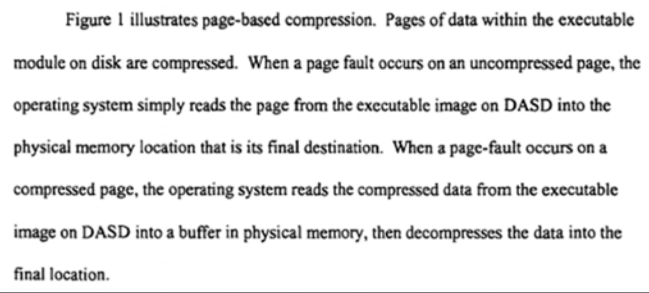
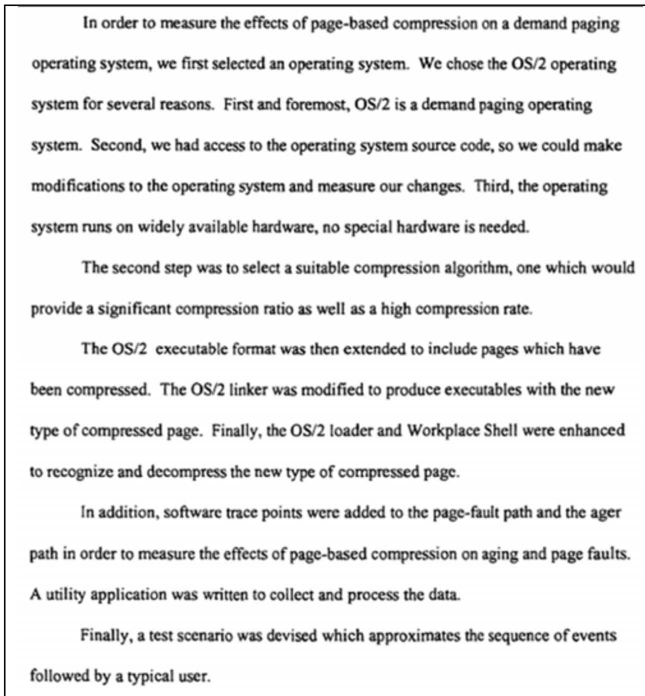


Figure 1 illustrates page-based compression. Pages of data within the executable module on disk are compressed. When a page fault occurs on an uncompressed page, the operating system simply reads the page from the executable image on DASD into the physical memory location that is its final destination. When a page-fault occurs on a compressed page, the operating system reads the compressed data from the executable image on DASD into a buffer in physical memory, then decompresses the data into the final location.



In order to measure the effects of page-based compression on a demand paging operating system, we first selected an operating system. We chose the OS/2 operating system for several reasons. First and foremost, OS/2 is a demand paging operating system. Second, we had access to the operating system source code, so we could make modifications to the operating system and measure our changes. Third, the operating system runs on widely available hardware, no special hardware is needed.

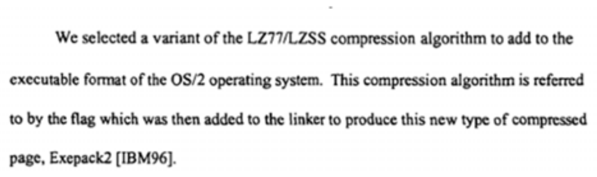
The second step was to select a suitable compression algorithm, one which would provide a significant compression ratio as well as a high compression rate.

The OS/2 executable format was then extended to include pages which have been compressed. The OS/2 linker was modified to produce executables with the new type of compressed page. Finally, the OS/2 loader and Workplace Shell were enhanced to recognize and decompress the new type of compressed page.

In addition, software trace points were added to the page-fault path and the ager path in order to measure the effects of page-based compression on aging and page faults. A utility application was written to collect and process the data.

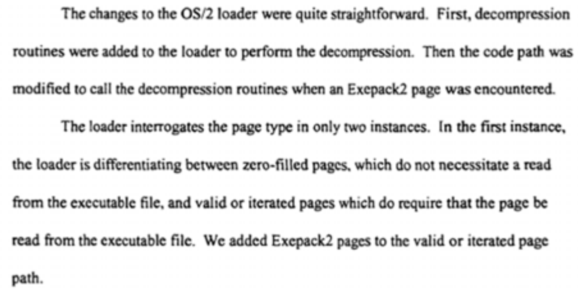
Finally, a test scenario was devised which approximates the sequence of events followed by a typical user.

Wynn Thesis, 5.1



We selected a variant of the LZ77/LZSS compression algorithm to add to the executable format of the OS/2 operating system. This compression algorithm is referred to by the flag which was then added to the linker to produce this new type of compressed page, Exepack2 [IBM96].

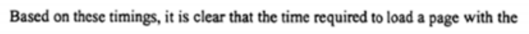
Wynn Thesis, 5.2



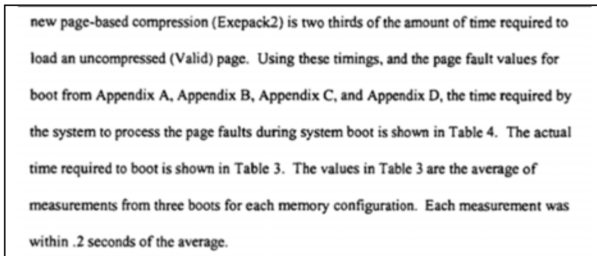
The changes to the OS/2 loader were quite straightforward. First, decompression routines were added to the loader to perform the decompression. Then the code path was modified to call the decompression routines when an Exepack2 page was encountered.

The loader interrogates the page type in only two instances. In the first instance, the loader is differentiating between zero-filled pages, which do not necessitate a read from the executable file, and valid or iterated pages which do require that the page be read from the executable file. We added Exepack2 pages to the valid or iterated page path.

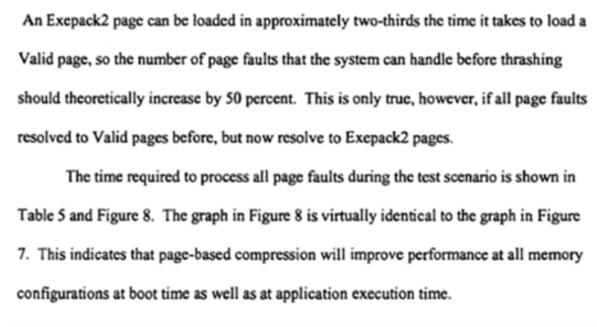
Wynn Thesis, 5.5



Based on these timings, it is clear that the time required to load a page with the

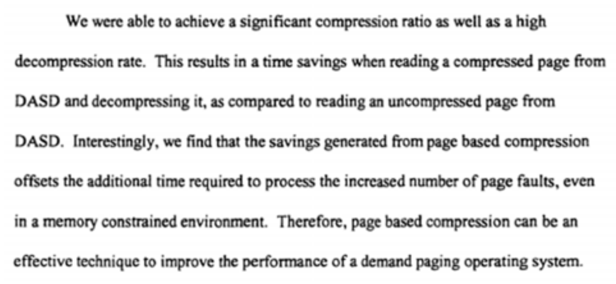

new page-based compression (Exepack2) is two thirds of the amount of time required to load an uncompressed (Valid) page. Using these timings, and the page fault values for boot from Appendix A, Appendix B, Appendix C, and Appendix D, the time required by the system to process the page faults during system boot is shown in Table 4. The actual time required to boot is shown in Table 3. The values in Table 3 are the average of measurements from three boots for each memory configuration. Each measurement was within .2 seconds of the average.

Wynn Thesis, 6.3


An Exepack2 page can be loaded in approximately two-thirds the time it takes to load a Valid page, so the number of page faults that the system can handle before thrashing should theoretically increase by 50 percent. This is only true, however, if all page faults resolved to Valid pages before, but now resolve to Exepack2 pages.

The time required to process all page faults during the test scenario is shown in Table 5 and Figure 8. The graph in Figure 8 is virtually identical to the graph in Figure 7. This indicates that page-based compression will improve performance at all memory configurations at boot time as well as at application execution time.

Wynn Thesis, 6.3


We were able to achieve a significant compression ratio as well as a high decompression rate. This results in a time savings when reading a compressed page from DASD and decompressing it, as compared to reading an uncompressed page from DASD. Interestingly, we find that the savings generated from page based compression offsets the additional time required to process the increased number of page faults, even in a memory constrained environment. Therefore, page based compression can be an effective technique to improve the performance of a demand paging operating system.

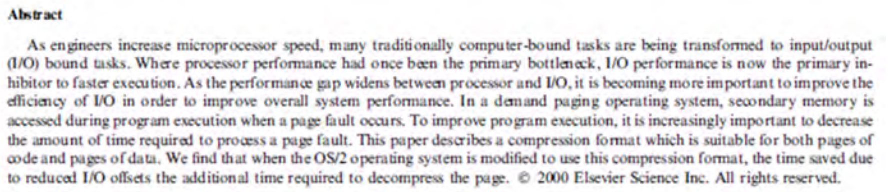
Wynn Thesis, 7.2

See also Wynn Thesis, Table 2-4, Fig. 7

1.4 accessing and decompressing said at least a portion of said boot data in said compressed form from said memory; and	Wynn, as evidenced by the example citations below, discloses "accessing and decompressing said at least a portion of said boot data in said compressed form from said memory; and"

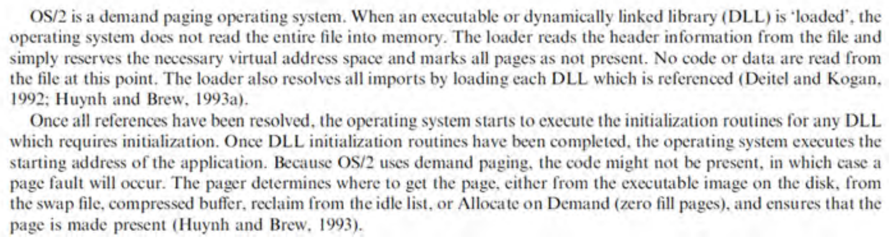
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing and decompressing said at least a portion of said boot data in said compressed form from said memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Wynn discloses this limitation:


Abstract

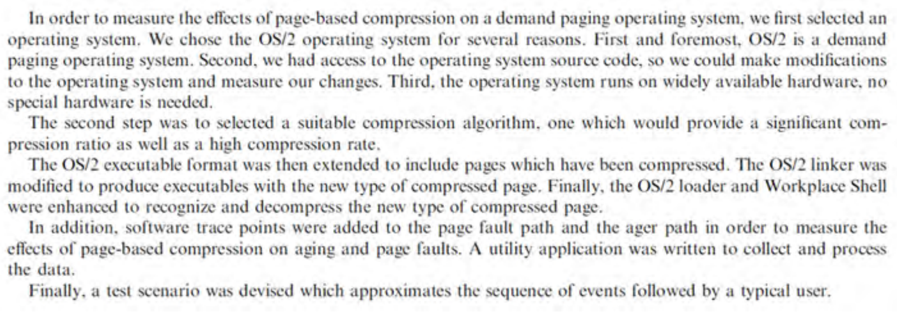
As engineers increase microprocessor speed, many traditionally computer-bound tasks are being transformed to input/output (I/O) bound tasks. Where processor performance had once been the primary bottleneck, I/O performance is now the primary inhibitor to faster execution. As the performance gap widens between processor and I/O, it is becoming more important to improve the efficiency of I/O in order to improve overall system performance. In a demand paging operating system, secondary memory is accessed during program execution when a page fault occurs. To improve program execution, it is increasingly important to decrease the amount of time required to process a page fault. This paper describes a compression format which is suitable for both pages of code and pages of data. We find that when the OS/2 operating system is modified to use this compression format, the time saved due to reduced I/O offsets the additional time required to decompress the page. © 2000 Elsevier Science Inc. All rights reserved.

Wynn Article, Abstract


OS/2 is a demand paging operating system. When an executable or dynamically linked library (DLL) is 'loaded', the operating system does not read the entire file into memory. The loader reads the header information from the file and simply reserves the necessary virtual address space and marks all pages as not present. No code or data are read from the file at this point. The loader also resolves all imports by loading each DLL which is referenced (Deitel and Kogan, 1992; Huynh and Brew, 1993a).

Once all references have been resolved, the operating system starts to execute the initialization routines for any DLL which requires initialization. Once DLL initialization routines have been completed, the operating system executes the starting address of the application. Because OS/2 uses demand paging, the code might not be present, in which case a page fault will occur. The pager determines where to get the page, either from the executable image on the disk, from the swap file, compressed buffer, reclaim from the idle list, or Allocate on Demand (zero fill pages), and ensures that the page is made present (Huynh and Brew, 1993).

Wynn Article, 2.2


In order to measure the effects of page-based compression on a demand paging operating system, we first selected an operating system. We chose the OS/2 operating system for several reasons. First and foremost, OS/2 is a demand paging operating system. Second, we had access to the operating system source code, so we could make modifications to the operating system and measure our changes. Third, the operating system runs on widely available hardware, no special hardware is needed.

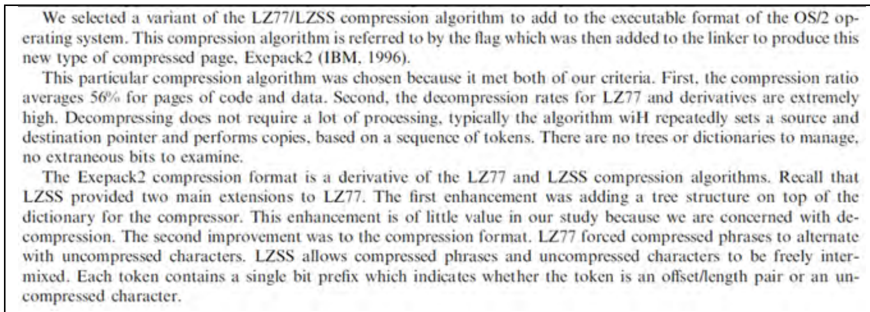
The second step was to selected a suitable compression algorithm, one which would provide a significant compression ratio as well as a high compression rate.

The OS/2 executable format was then extended to include pages which have been compressed. The OS/2 linker was modified to produce executables with the new type of compressed page. Finally, the OS/2 loader and Workplace Shell were enhanced to recognize and decompress the new type of compressed page.

In addition, software trace points were added to the page fault path and the ager path in order to measure the effects of page-based compression on aging and page faults. A utility application was written to collect and process the data.

Finally, a test scenario was devised which approximates the sequence of events followed by a typical user.

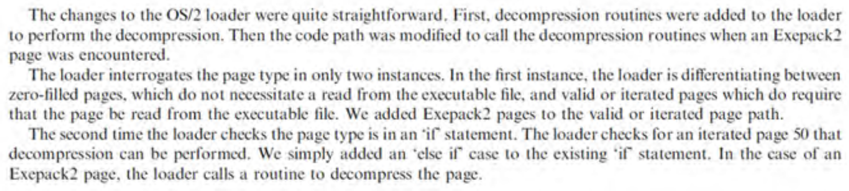
Wynn Article, 3.1


We selected a variant of the LZ77/LZSS compression algorithm to add to the executable format of the OS/2 operating system. This compression algorithm is referred to by the flag which was then added to the linker to produce this new type of compressed page, Exepack2 (IBM, 1996).

This particular compression algorithm was chosen because it met both of our criteria. First, the compression ratio averages 56% for pages of code and data. Second, the decompression rates for LZ77 and derivatives are extremely high. Decompressing does not require a lot of processing, typically the algorithm wiH repeatedly sets a source and destination pointer and performs copies, based on a sequence of tokens. There are no trees or dictionaries to manage, no extraneous bits to examine.

The Exepack2 compression format is a derivative of the LZ77 and LZSS compression algorithms. Recall that LZSS provided two main extensions to LZ77. The first enhancement was adding a tree structure on top of the dictionary for the compressor. This enhancement is of little value in our study because we are concerned with decompression. The second improvement was to the compression format. LZ77 forced compressed phrases to alternate with uncompressed characters. LZSS allows compressed phrases and uncompressed characters to be freely intermixed. Each token contains a single bit prefix which indicates whether the token is an offset/length pair or an uncompressed character.

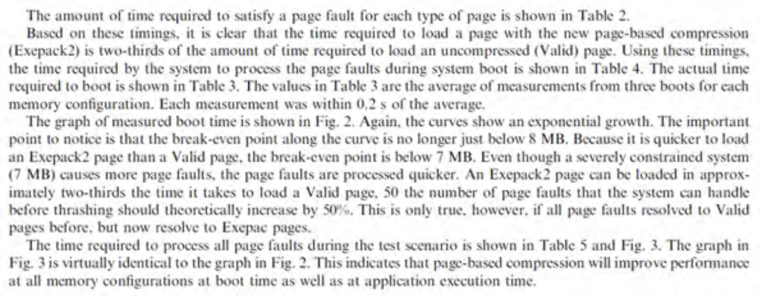
Wynn Article, 3.2


The changes to the OS/2 loader were quite straightforward. First, decompression routines were added to the loader to perform the decompression. Then the code path was modified to call the decompression routines when an Exepack2 page was encountered.

The loader interrogates the page type in only two instances. In the first instance, the loader is differentiating between zero-filled pages, which do not necessitate a read from the executable file, and valid or iterated pages which do require that the page be read from the executable file. We added Exepack2 pages to the valid or iterated page path.

The second time the loader checks the page type is in an 'if' statement. The loader checks for an iterated page 50 that decompression can be performed. We simply added an 'else if' case to the existing 'if' statement. In the case of an Exepack2 page, the loader calls a routine to decompress the page.

Wynn Article, 3.4


The amount of time required to satisfy a page fault for each type of page is shown in Table 2.

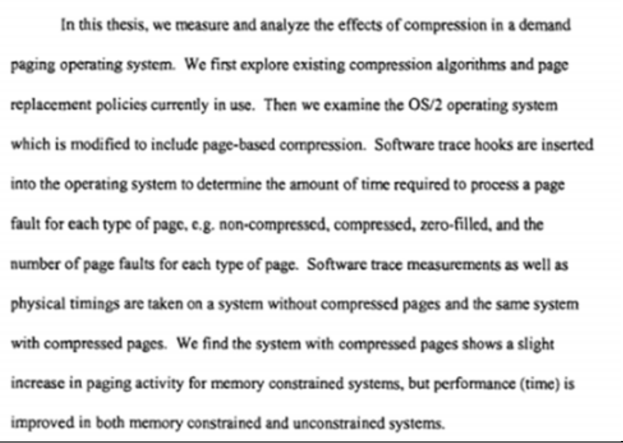
Based on these timings, it is clear that the time required to load a page with the new page-based compression (Exepack2) is two-thirds of the amount of time required to load an uncompressed (Valid) page. Using these timings, the time required by the system to process the page faults during system boot is shown in Table 4. The actual time required to boot is shown in Table 3. The values in Table 3 are the average of measurements from three boots for each memory configuration. Each measurement was within 0.2 s of the average.

The graph of measured boot time is shown in Fig. 2. Again, the curves show an exponential growth. The important point to notice is that the break-even point along the curve is no longer just below 8 MB. Because it is quicker to load an Exepack2 page than a Valid page, the break-even point is below 7 MB. Even though a severely constrained system (7 MB) causes more page faults, the page faults are processed quicker. An Exepack2 page can be loaded in approximately two-thirds the time it takes to load a Valid page, 50 the number of page faults that the system can handle before thrashing should theoretically increase by 50%. This is only true, however, if all page faults resolved to Valid pages before, but now resolve to Exepac pages.

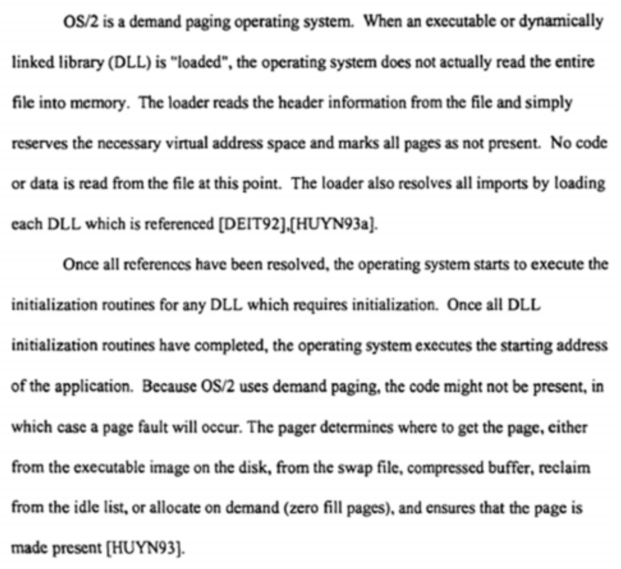
The time required to process all page faults during the test scenario is shown in Table 5 and Fig. 3. The graph in Fig. 3 is virtually identical to the graph in Fig. 2. This indicates that page-based compression will improve performance at all memory configurations at boot time as well as at application execution time.

Wynn Article, 4.3

See also, Wynn Article, Table 3 and Fig. 2


In this thesis, we measure and analyze the effects of compression in a demand paging operating system. We first explore existing compression algorithms and page replacement policies currently in use. Then we examine the OS/2 operating system which is modified to include page-based compression. Software trace hooks are inserted into the operating system to determine the amount of time required to process a page fault for each type of page, e.g. non-compressed, compressed, zero-filled, and the number of page faults for each type of page. Software trace measurements as well as physical timings are taken on a system without compressed pages and the same system with compressed pages. We find the system with compressed pages shows a slight increase in paging activity for memory constrained systems, but performance (time) is improved in both memory constrained and unconstrained systems.

Wynn Thesis, Abstract



OS/2 is a demand paging operating system. When an executable or dynamically linked library (DLL) is "loaded", the operating system does not actually read the entire file into memory. The loader reads the header information from the file and simply reserves the necessary virtual address space and marks all pages as not present. No code or data is read from the file at this point. The loader also resolves all imports by loading each DLL which is referenced [DEIT92],[HUYN93a].

Once all references have been resolved, the operating system starts to execute the initialization routines for any DLL which requires initialization. Once all DLL initialization routines have completed, the operating system executes the starting address of the application. Because OS/2 uses demand paging, the code might not be present, in which case a page fault will occur. The pager determines where to get the page, either from the executable image on the disk, from the swap file, compressed buffer, reclaim from the idle list, or allocate on demand (zero fill pages), and ensures that the page is made present [HUYN93].

Wynn Thesis, 4.2

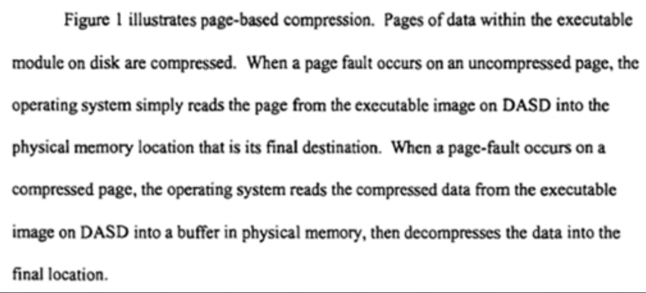
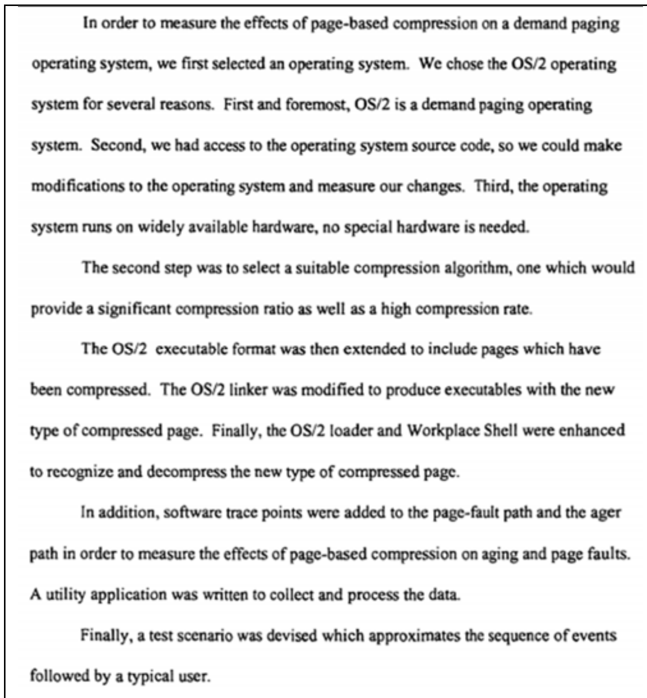


Figure 1 illustrates page-based compression. Pages of data within the executable module on disk are compressed. When a page fault occurs on an uncompressed page, the operating system simply reads the page from the executable image on DASD into the physical memory location that is its final destination. When a page-fault occurs on a compressed page, the operating system reads the compressed data from the executable image on DASD into a buffer in physical memory, then decompresses the data into the final location.


In order to measure the effects of page-based compression on a demand paging operating system, we first selected an operating system. We chose the OS/2 operating system for several reasons. First and foremost, OS/2 is a demand paging operating system. Second, we had access to the operating system source code, so we could make modifications to the operating system and measure our changes. Third, the operating system runs on widely available hardware, no special hardware is needed.

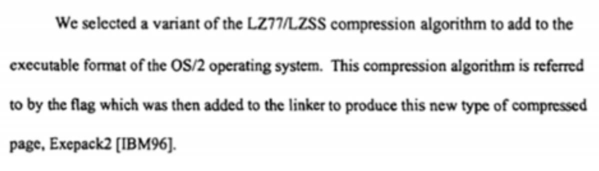
The second step was to select a suitable compression algorithm, one which would provide a significant compression ratio as well as a high compression rate.

The OS/2 executable format was then extended to include pages which have been compressed. The OS/2 linker was modified to produce executables with the new type of compressed page. Finally, the OS/2 loader and Workplace Shell were enhanced to recognize and decompress the new type of compressed page.

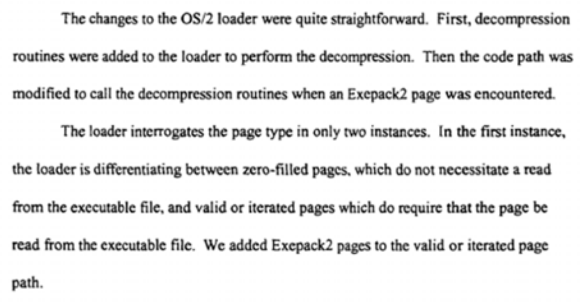
In addition, software trace points were added to the page-fault path and the ager path in order to measure the effects of page-based compression on aging and page faults. A utility application was written to collect and process the data.

Finally, a test scenario was devised which approximates the sequence of events followed by a typical user.

Wynn Thesis, 5.1

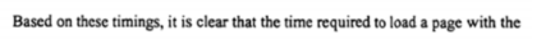

We selected a variant of the LZ77/LZSS compression algorithm to add to the executable format of the OS/2 operating system. This compression algorithm is referred to by the flag which was then added to the linker to produce this new type of compressed page, Exepack2 [IBM96].

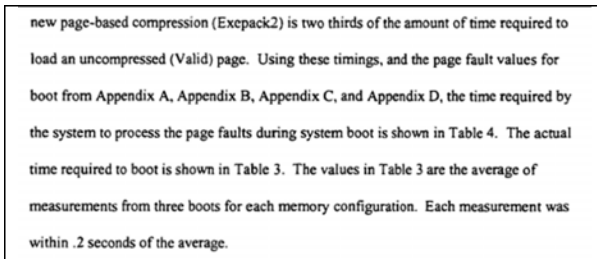
Wynn Thesis, 5.2


The changes to the OS/2 loader were quite straightforward. First, decompression routines were added to the loader to perform the decompression. Then the code path was modified to call the decompression routines when an Exepack2 page was encountered.

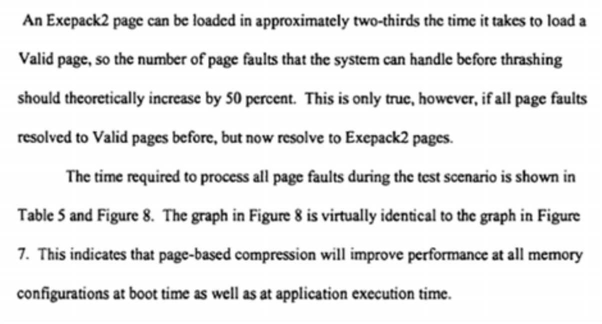
The loader interrogates the page type in only two instances. In the first instance, the loader is differentiating between zero-filled pages, which do not necessitate a read from the executable file, and valid or iterated pages which do require that the page be read from the executable file. We added Exepack2 pages to the valid or iterated page path.

Wynn Thesis, 5.5


Based on these timings, it is clear that the time required to load a page with the

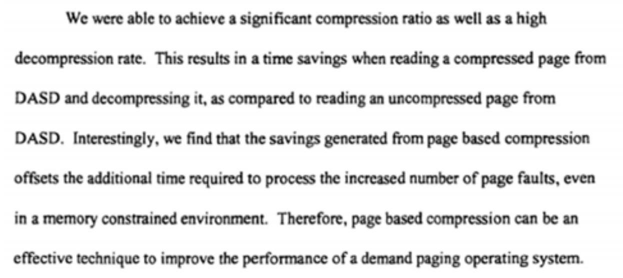

new page-based compression (Exepack2) is two thirds of the amount of time required to load an uncompressed (Valid) page. Using these timings, and the page fault values for boot from Appendix A, Appendix B, Appendix C, and Appendix D, the time required by the system to process the page faults during system boot is shown in Table 4. The actual time required to boot is shown in Table 3. The values in Table 3 are the average of measurements from three boots for each memory configuration. Each measurement was within .2 seconds of the average.

Wynn Thesis, 6.3


An Exepack2 page can be loaded in approximately two-thirds the time it takes to load a Valid page, so the number of page faults that the system can handle before thrashing should theoretically increase by 50 percent. This is only true, however, if all page faults resolved to Valid pages before, but now resolve to Exepack2 pages.

The time required to process all page faults during the test scenario is shown in Table 5 and Figure 8. The graph in Figure 8 is virtually identical to the graph in Figure 7. This indicates that page-based compression will improve performance at all memory configurations at boot time as well as at application execution time.

Wynn Thesis, 6.3


We were able to achieve a significant compression ratio as well as a high decompression rate. This results in a time savings when reading a compressed page from DASD and decompressing it, as compared to reading an uncompressed page from DASD. Interestingly, we find that the savings generated from page based compression offsets the additional time required to process the increased number of page faults, even in a memory constrained environment. Therefore, page based compression can be an effective technique to improve the performance of a demand paging operating system.

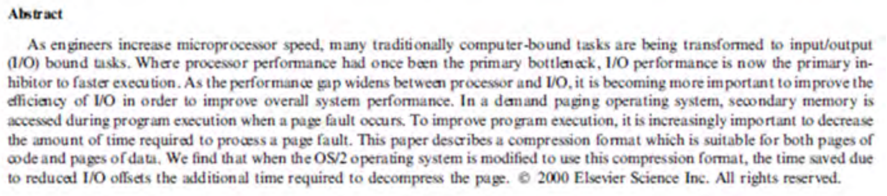
Wynn Thesis, 7.2

See also Wynn Thesis, Table 2-4, Fig. 7

1.5 utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine.	Wynn, as evidenced by the example citations below, discloses "utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

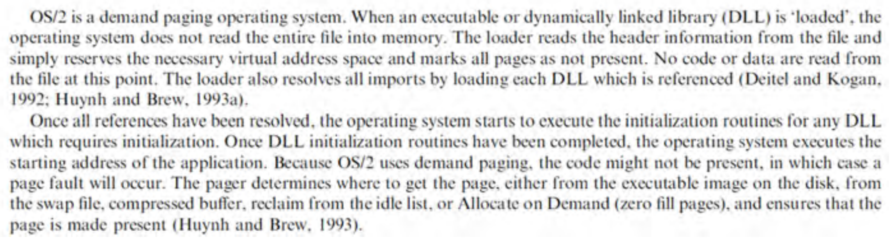
Wynn discloses this limitation:



Abstract

As engineers increase microprocessor speed, many traditionally computer-bound tasks are being transformed to input/output (I/O) bound tasks. Where processor performance had once been the primary bottleneck, I/O performance is now the primary inhibitor to faster execution. As the performance gap widens between processor and I/O, it is becoming more important to improve the efficiency of I/O in order to improve overall system performance. In a demand paging operating system, secondary memory is accessed during program execution when a page fault occurs. To improve program execution, it is increasingly important to decrease the amount of time required to process a page fault. This paper describes a compression format which is suitable for both pages of code and pages of data. We find that when the OS/2 operating system is modified to use this compression format, the time saved due to reduced I/O offsets the additional time required to decompress the page. © 2000 Elsevier Science Inc. All rights reserved.

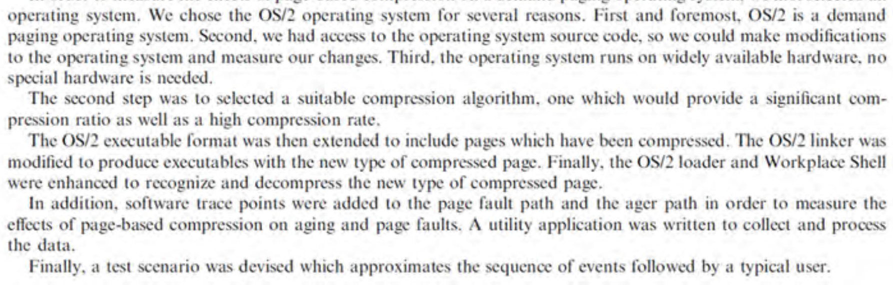
Wynn Article, Abstract



OS/2 is a demand paging operating system. When an executable or dynamically linked library (DLL) is 'loaded', the operating system does not read the entire file into memory. The loader reads the header information from the file and simply reserves the necessary virtual address space and marks all pages as not present. No code or data are read from the file at this point. The loader also resolves all imports by loading each DLL which is referenced (Deitel and Kogan, 1992; Huynh and Brew, 1993a).

Once all references have been resolved, the operating system starts to execute the initialization routines for any DLL which requires initialization. Once DLL initialization routines have been completed, the operating system executes the starting address of the application. Because OS/2 uses demand paging, the code might not be present, in which case a page fault will occur. The pager determines where to get the page, either from the executable image on the disk, from the swap file, compressed buffer, reclaim from the idle list, or Allocate on Demand (zero fill pages), and ensures that the page is made present (Huynh and Brew, 1993).

Wynn Article, 2.2



In order to measure the effects of page-based compression on a demand paging operating system, we first selected an operating system. We chose the OS/2 operating system for several reasons. First and foremost, OS/2 is a demand paging operating system. Second, we had access to the operating system source code, so we could make modifications to the operating system and measure our changes. Third, the operating system runs on widely available hardware, no special hardware is needed.

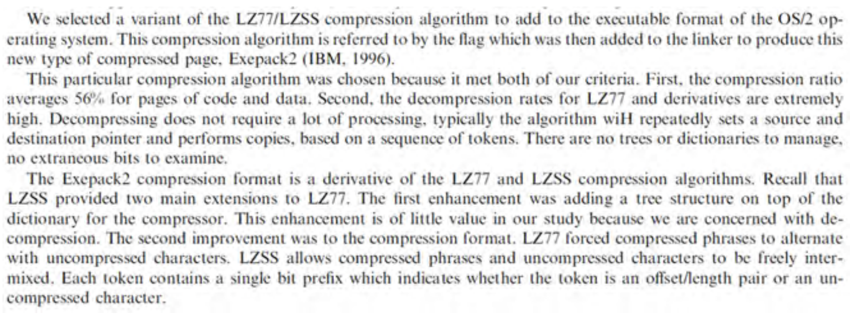
The second step was to selected a suitable compression algorithm, one which would provide a significant compression ratio as well as a high compression rate.

The OS/2 executable format was then extended to include pages which have been compressed. The OS/2 linker was modified to produce executables with the new type of compressed page. Finally, the OS/2 loader and Workplace Shell were enhanced to recognize and decompress the new type of compressed page.

In addition, software trace points were added to the page fault path and the ager path in order to measure the effects of page-based compression on aging and page faults. A utility application was written to collect and process the data.

Finally, a test scenario was devised which approximates the sequence of events followed by a typical user.

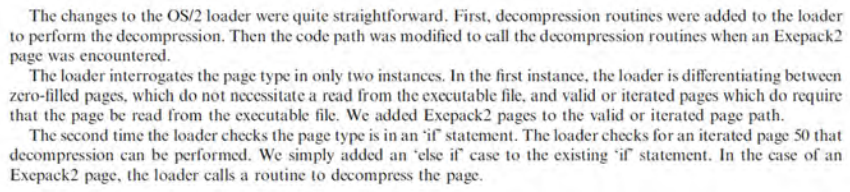
Wynn Article, 3.1


We selected a variant of the LZ77/LZSS compression algorithm to add to the executable format of the OS/2 operating system. This compression algorithm is referred to by the flag which was then added to the linker to produce this new type of compressed page, Exepack2 (IBM, 1996).

This particular compression algorithm was chosen because it met both of our criteria. First, the compression ratio averages 56% for pages of code and data. Second, the decompression rates for LZ77 and derivatives are extremely high. Decompressing does not require a lot of processing, typically the algorithm wiH repeatedly sets a source and destination pointer and performs copies, based on a sequence of tokens. There are no trees or dictionaries to manage, no extraneous bits to examine.

The Exepack2 compression format is a derivative of the LZ77 and LZSS compression algorithms. Recall that LZSS provided two main extensions to LZ77. The first enhancement was adding a tree structure on top of the dictionary for the compressor. This enhancement is of little value in our study because we are concerned with decompression. The second improvement was to the compression format. LZ77 forced compressed phrases to alternate with uncompressed characters. LZSS allows compressed phrases and uncompressed characters to be freely intermixed. Each token contains a single bit prefix which indicates whether the token is an offset/length pair or an uncompressed character.

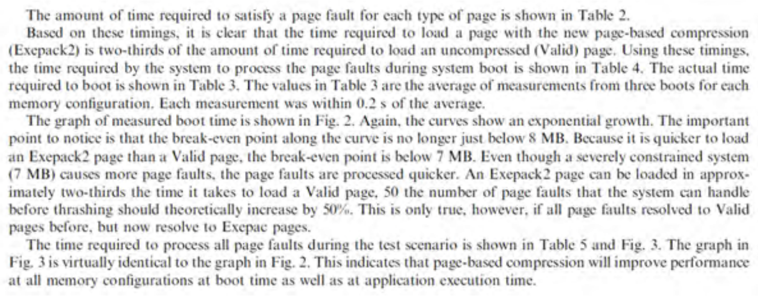
Wynn Article, 3.2


The changes to the OS/2 loader were quite straightforward. First, decompression routines were added to the loader to perform the decompression. Then the code path was modified to call the decompression routines when an Exepack2 page was encountered.

The loader interrogates the page type in only two instances. In the first instance, the loader is differentiating between zero-filled pages, which do not necessitate a read from the executable file, and valid or iterated pages which do require that the page be read from the executable file. We added Exepack2 pages to the valid or iterated page path.

The second time the loader checks the page type is in an 'if' statement. The loader checks for an iterated page 50 that decompression can be performed. We simply added an 'else if' case to the existing 'if' statement. In the case of an Exepack2 page, the loader calls a routine to decompress the page.

Wynn Article, 3.4


The amount of time required to satisfy a page fault for each type of page is shown in Table 2.

Based on these timings, it is clear that the time required to load a page with the new page-based compression (Exepack2) is two-thirds of the amount of time required to load an uncompressed (Valid) page. Using these timings, the time required by the system to process the page faults during system boot is shown in Table 4. The actual time required to boot is shown in Table 3. The values in Table 3 are the average of measurements from three boots for each memory configuration. Each measurement was within 0.2 s of the average.

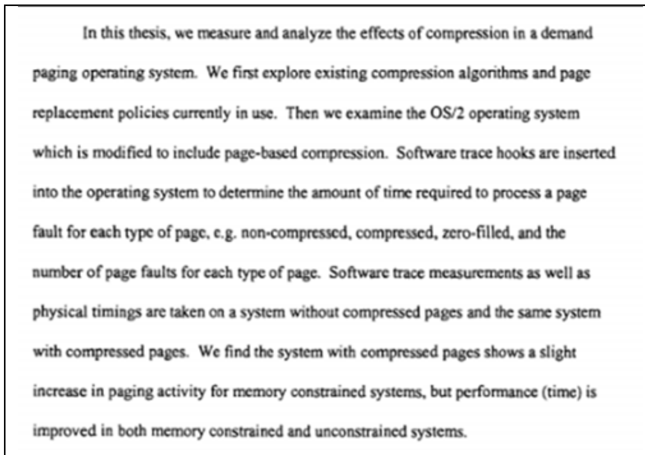
The graph of measured boot time is shown in Fig. 2. Again, the curves show an exponential growth. The important point to notice is that the break-even point along the curve is no longer just below 8 MB. Because it is quicker to load an Exepack2 page than a Valid page, the break-even point is below 7 MB. Even though a severely constrained system (7 MB) causes more page faults, the page faults are processed quicker. An Exepack2 page can be loaded in approximately two-thirds the time it takes to load a Valid page, 50 the number of page faults that the system can handle before thrashing should theoretically increase by 50%. This is only true, however, if all page faults resolved to Valid pages before, but now resolve to Exepac pages.

The time required to process all page faults during the test scenario is shown in Table 5 and Fig. 3. The graph in Fig. 3 is virtually identical to the graph in Fig. 2. This indicates that page-based compression will improve performance at all memory configurations at boot time as well as at application execution time.

Wynn Article, 4.3

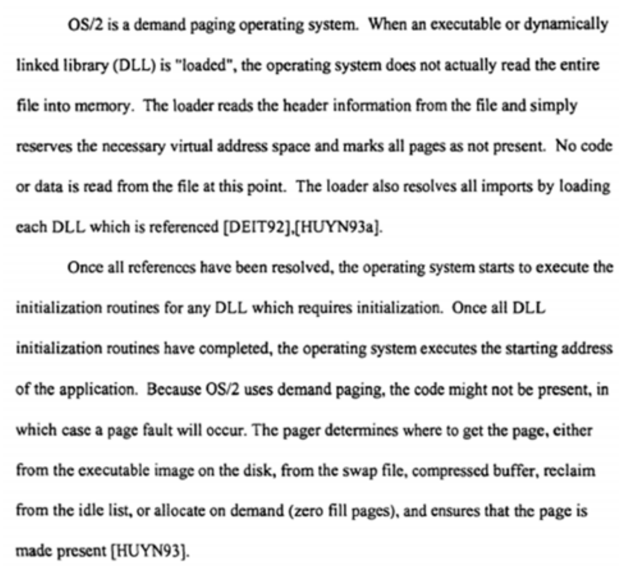
See also, Wynn Article, Table 3 and Fig. 2

Appendix B37
Invalidity of U.S. Patent 8,090,936 based on Wynn



In this thesis, we measure and analyze the effects of compression in a demand paging operating system. We first explore existing compression algorithms and page replacement policies currently in use. Then we examine the OS/2 operating system which is modified to include page-based compression. Software trace hooks are inserted into the operating system to determine the amount of time required to process a page fault for each type of page, e.g. non-compressed, compressed, zero-filled, and the number of page faults for each type of page. Software trace measurements as well as physical timings are taken on a system without compressed pages and the same system with compressed pages. We find the system with compressed pages shows a slight increase in paging activity for memory constrained systems, but performance (time) is improved in both memory constrained and unconstrained systems.

Wynn Thesis, Abstract



OS/2 is a demand paging operating system. When an executable or dynamically linked library (DLL) is "loaded", the operating system does not actually read the entire file into memory. The loader reads the header information from the file and simply reserves the necessary virtual address space and marks all pages as not present. No code or data is read from the file at this point. The loader also resolves all imports by loading each DLL which is referenced [DEIT92],[HUYN93a].

Once all references have been resolved, the operating system starts to execute the initialization routines for any DLL which requires initialization. Once all DLL initialization routines have completed, the operating system executes the starting address of the application. Because OS/2 uses demand paging, the code might not be present, in which case a page fault will occur. The pager determines where to get the page, either from the executable image on the disk, from the swap file, compressed buffer, reclaim from the idle list, or allocate on demand (zero fill pages), and ensures that the page is made present [HUYN93].

Wynn Thesis, 4.2

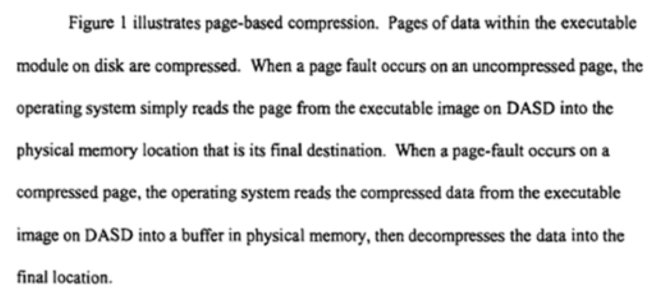
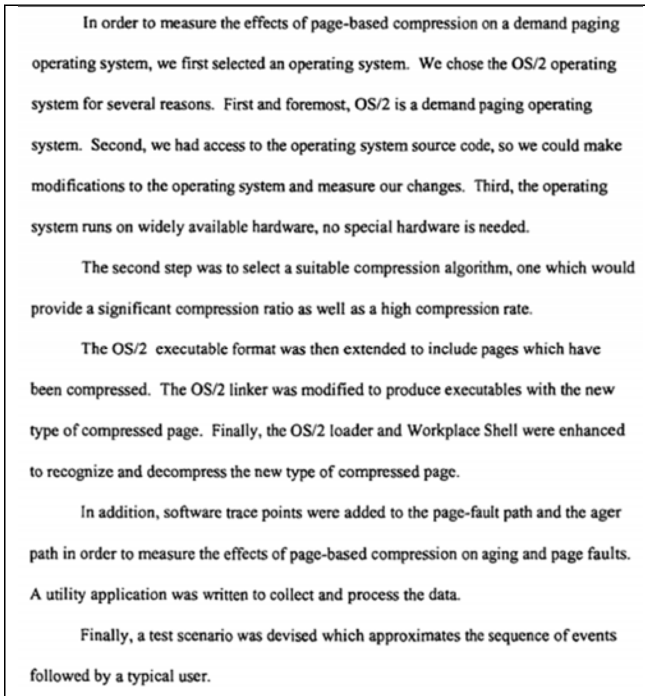


Figure 1 illustrates page-based compression. Pages of data within the executable module on disk are compressed. When a page fault occurs on an uncompressed page, the operating system simply reads the page from the executable image on DASD into the physical memory location that is its final destination. When a page-fault occurs on a compressed page, the operating system reads the compressed data from the executable image on DASD into a buffer in physical memory, then decompresses the data into the final location.


In order to measure the effects of page-based compression on a demand paging operating system, we first selected an operating system. We chose the OS/2 operating system for several reasons. First and foremost, OS/2 is a demand paging operating system. Second, we had access to the operating system source code, so we could make modifications to the operating system and measure our changes. Third, the operating system runs on widely available hardware, no special hardware is needed.

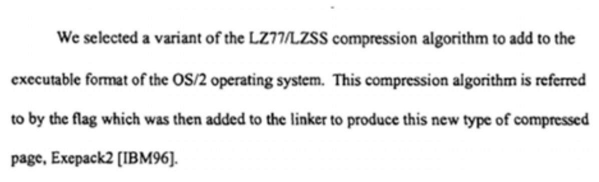
The second step was to select a suitable compression algorithm, one which would provide a significant compression ratio as well as a high compression rate.

The OS/2 executable format was then extended to include pages which have been compressed. The OS/2 linker was modified to produce executables with the new type of compressed page. Finally, the OS/2 loader and Workplace Shell were enhanced to recognize and decompress the new type of compressed page.

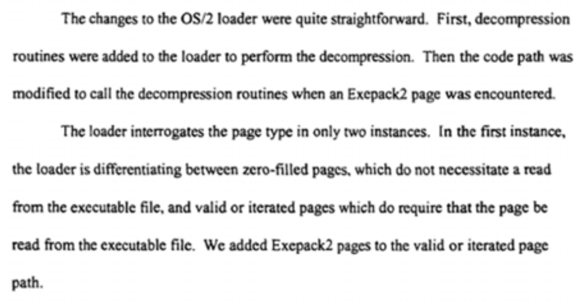
In addition, software trace points were added to the page-fault path and the ager path in order to measure the effects of page-based compression on aging and page faults. A utility application was written to collect and process the data.

Finally, a test scenario was devised which approximates the sequence of events followed by a typical user.

Wynn Thesis, 5.1

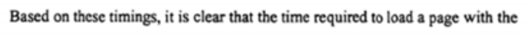

We selected a variant of the LZ77/LZSS compression algorithm to add to the executable format of the OS/2 operating system. This compression algorithm is referred to by the flag which was then added to the linker to produce this new type of compressed page, Exepack2 [IBM96].

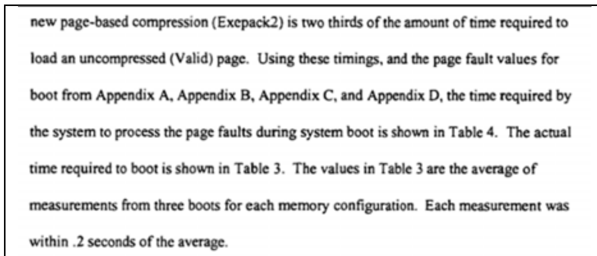
Wynn Thesis, 5.2


The changes to the OS/2 loader were quite straightforward. First, decompression routines were added to the loader to perform the decompression. Then the code path was modified to call the decompression routines when an Exepack2 page was encountered.

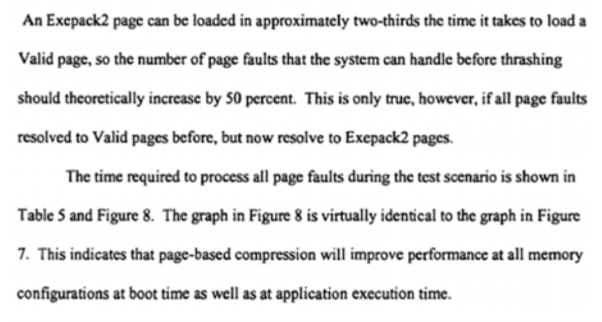
The loader interrogates the page type in only two instances. In the first instance, the loader is differentiating between zero-filled pages, which do not necessitate a read from the executable file, and valid or iterated pages which do require that the page be read from the executable file. We added Exepack2 pages to the valid or iterated page path.

Wynn Thesis, 5.5


Based on these timings, it is clear that the time required to load a page with the

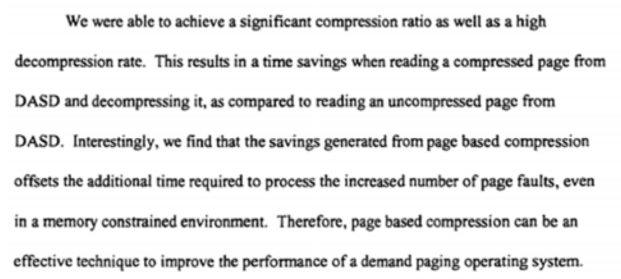

new page-based compression (Exepack2) is two thirds of the amount of time required to load an uncompressed (Valid) page. Using these timings, and the page fault values for boot from Appendix A, Appendix B, Appendix C, and Appendix D, the time required by the system to process the page faults during system boot is shown in Table 4. The actual time required to boot is shown in Table 3. The values in Table 3 are the average of measurements from three boots for each memory configuration. Each measurement was within .2 seconds of the average.

Wynn Thesis, 6.3


An Exepack2 page can be loaded in approximately two-thirds the time it takes to load a Valid page, so the number of page faults that the system can handle before thrashing should theoretically increase by 50 percent. This is only true, however, if all page faults resolved to Valid pages before, but now resolve to Exepack2 pages.

The time required to process all page faults during the test scenario is shown in Table 5 and Figure 8. The graph in Figure 8 is virtually identical to the graph in Figure 7. This indicates that page-based compression will improve performance at all memory configurations at boot time as well as at application execution time.

Wynn Thesis, 6.3


We were able to achieve a significant compression ratio as well as a high decompression rate. This results in a time savings when reading a compressed page from DASD and decompressing it, as compared to reading an uncompressed page from DASD. Interestingly, we find that the savings generated from page based compression offsets the additional time required to process the increased number of page faults, even in a memory constrained environment. Therefore, page based compression can be an effective technique to improve the performance of a demand paging operating system.

Wynn Thesis, 7.2

See also Wynn Thesis, Table 2-4, Fig. 7

2. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system.	Wynn, as evidenced by the example citations below, discloses “said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Wynn discloses this limitation: *See* Claims 1.1, 1.4, and 1.5 above.	

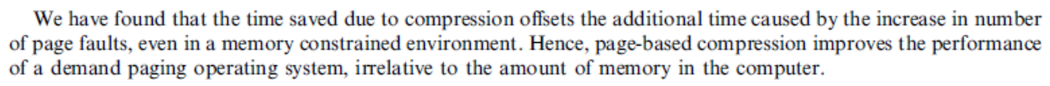
Appendix B37
Invalidity of U.S. Patent 8,090,936 based on Wynn

3. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system.	Wynn, as evidenced by the example citations below, discloses “said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system.”

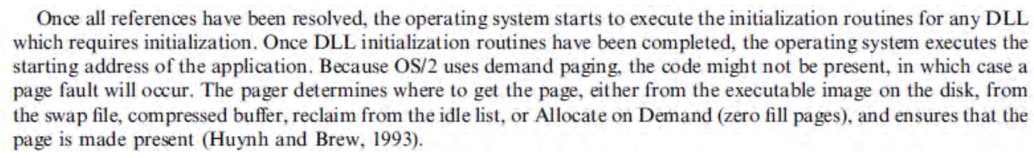
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.

Wynn discloses this limitation:

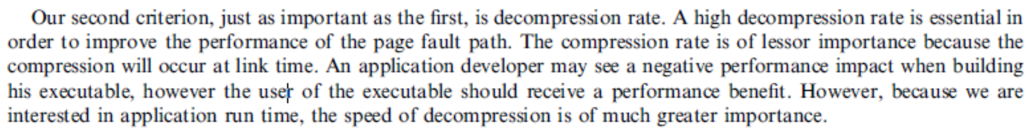
See Claims 1.1, 1.3, and 1.4 above.


We have found that the time saved due to compression offsets the additional time caused by the increase in number of page faults, even in a memory constrained environment. Hence, page-based compression improves the performance of a demand paging operating system, irrelative to the amount of memory in the computer.

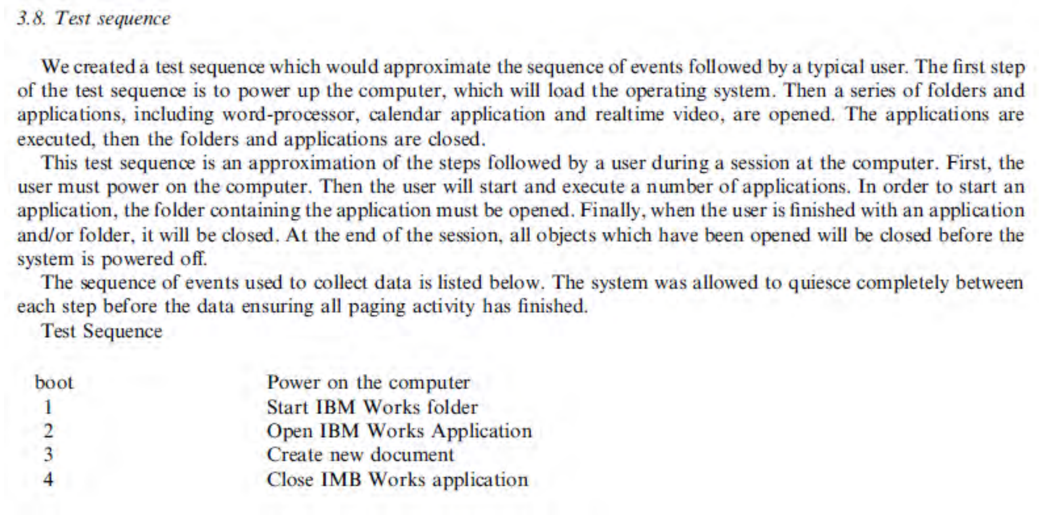
Wynn Article, 4.3


Once all references have been resolved, the operating system starts to execute the initialization routines for any DLL which requires initialization. Once DLL initialization routines have been completed, the operating system executes the starting address of the application. Because OS/2 uses demand paging, the code might not be present, in which case a page fault will occur. The pager determines where to get the page, either from the executable image on the disk, from the swap file, compressed buffer, reclaim from the idle list, or Allocate on Demand (zero fill pages), and ensures that the page is made present (Huynh and Brew, 1993).

Wynn Article, 2.2


Our second criterion, just as important as the first, is decompression rate. A high decompression rate is essential in order to improve the performance of the page fault path. The compression rate is of lessor importance because the compression will occur at link time. An application developer may see a negative performance impact when building his executable, however the user of the executable should receive a performance benefit. However, because we are interested in application run time, the speed of decompression is of much greater importance.

Wynn Article, 3.2



3.8. Test sequence

We created a test sequence which would approximate the sequence of events followed by a typical user. The first step of the test sequence is to power up the computer, which will load the operating system. Then a series of folders and applications, including word-processor, calendar application and realtime video, are opened. The applications are executed, then the folders and applications are closed.

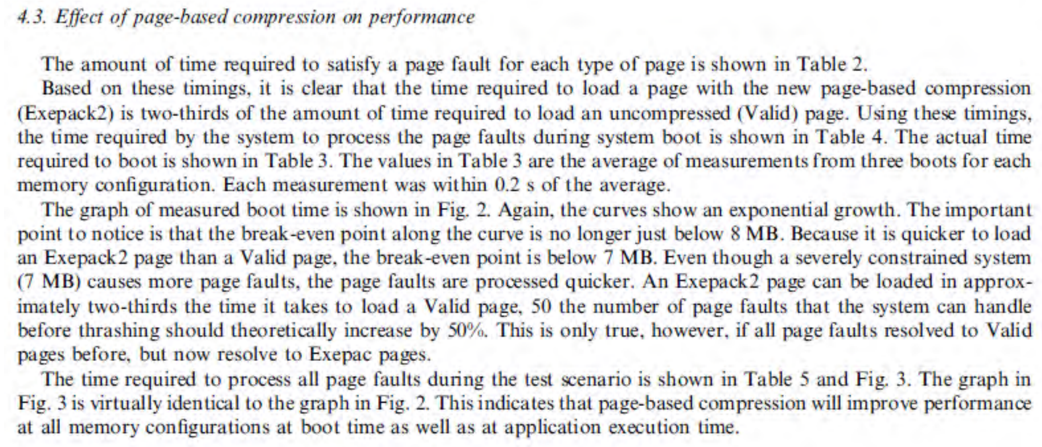
This test sequence is an approximation of the steps followed by a user during a session at the computer. First, the user must power on the computer. Then the user will start and execute a number of applications. In order to start an application, the folder containing the application must be opened. Finally, when the user is finished with an application and/or folder, it will be closed. At the end of the session, all objects which have been opened will be closed before the system is powered off.

The sequence of events used to collect data is listed below. The system was allowed to quiesce completely between each step before the data ensuring all paging activity has finished.

Test Sequence

boot	Power on the computer
1	Start IBM Works folder
2	Open IBM Works Application
3	Create new document
4	Close IMB Works application

Wynn Article, 3.8



4.3. Effect of page-based compression on performance

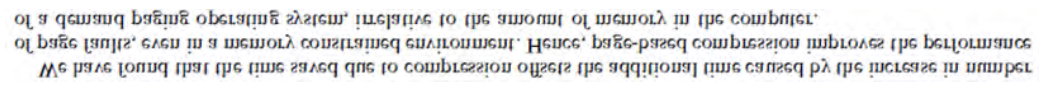
The amount of time required to satisfy a page fault for each type of page is shown in Table 2.

Based on these timings, it is clear that the time required to load a page with the new page-based compression (Exepack2) is two-thirds of the amount of time required to load an uncompressed (Valid) page. Using these timings, the time required by the system to process the page faults during system boot is shown in Table 4. The actual time required to boot is shown in Table 3. The values in Table 3 are the average of measurements from three boots for each memory configuration. Each measurement was within 0.2 s of the average.

The graph of measured boot time is shown in Fig. 2. Again, the curves show an exponential growth. The important point to notice is that the break-even point along the curve is no longer just below 8 MB. Because it is quicker to load an Exepack2 page than a Valid page, the break-even point is below 7 MB. Even though a severely constrained system (7 MB) causes more page faults, the page faults are processed quicker. An Exepack2 page can be loaded in approximately two-thirds the time it takes to load a Valid page, 50 the number of page faults that the system can handle before thrashing should theoretically increase by 50%. This is only true, however, if all page faults resolved to Valid pages before, but now resolve to Exepac pages.

The time required to process all page faults during the test scenario is shown in Table 5 and Fig. 3. The graph in Fig. 3 is virtually identical to the graph in Fig. 2. This indicates that page-based compression will improve performance at all memory configurations at boot time as well as at application execution time.

Wynn Article, 4.3



We have found that the time saved due to compression offsets the additional time caused by the increase in number of page faults, even in a memory constrained environment. Hence, page-based compression improves the performance of a demand paging operating system, irrelative to the amount of memory in the computer.

Wynn Article, 1.2

Applications load time is typically faster on operating systems that use virtual memory and demand paging. Present-day operating systems will, in general, not preload all of the code and data for an application at application load time. Smaller sections of code and data are loaded into memory as they are required by the application. The amount of code and data required at application load time is usually very much smaller than the total amount of code and data in the application. In fact, large sections of code and data may not be required by the application at all during a particular session.

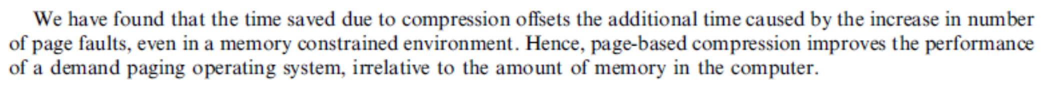
Wynn Thesis, 2.1

4. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an application program and an operating system of said computer system.	Wynn, as evidenced by the example citations below, discloses "said decompressed at least a portion of said boot data comprises program code associated with an application program and an operating system of said computer system."

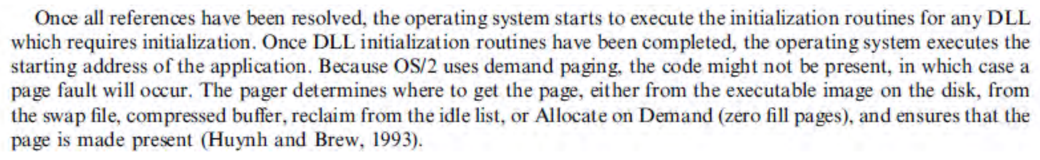
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Wynn discloses this limitation:

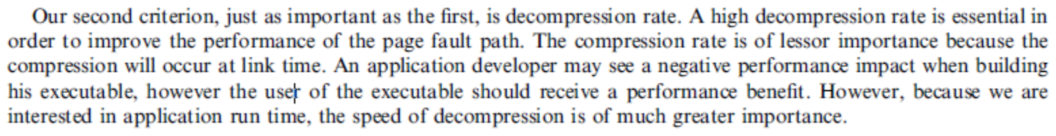
See Claims 1.1, 1.3, and 1.4 above.


We have found that the time saved due to compression offsets the additional time caused by the increase in number of page faults, even in a memory constrained environment. Hence, page-based compression improves the performance of a demand paging operating system, irrelative to the amount of memory in the computer.

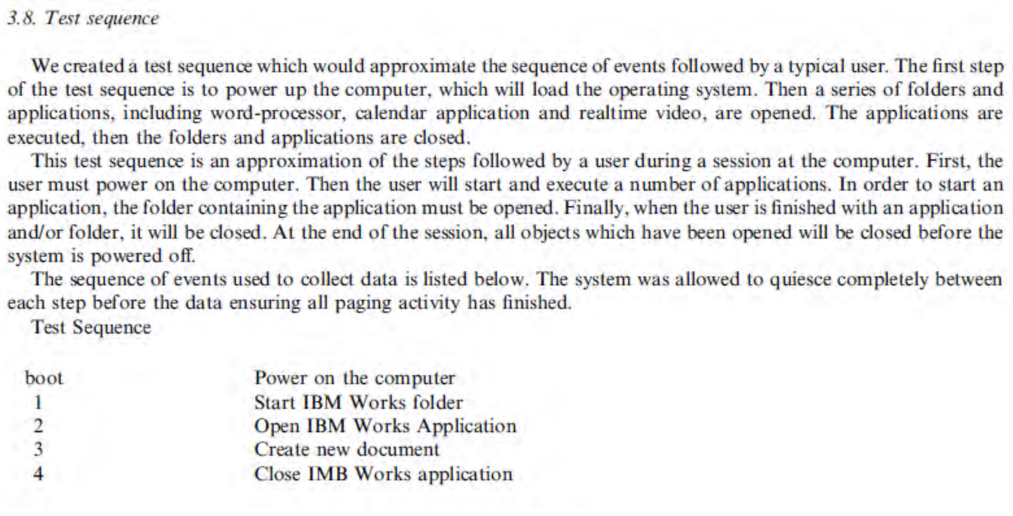
Wynn Article, 4.3


Once all references have been resolved, the operating system starts to execute the initialization routines for any DLL which requires initialization. Once DLL initialization routines have been completed, the operating system executes the starting address of the application. Because OS/2 uses demand paging, the code might not be present, in which case a page fault will occur. The pager determines where to get the page, either from the executable image on the disk, from the swap file, compressed buffer, reclaim from the idle list, or Allocate on Demand (zero fill pages), and ensures that the page is made present (Huynh and Brew, 1993).

Wynn Article, 2.2


Our second criterion, just as important as the first, is decompression rate. A high decompression rate is essential in order to improve the performance of the page fault path. The compression rate is of lessor importance because the compression will occur at link time. An application developer may see a negative performance impact when building his executable, however the user of the executable should receive a performance benefit. However, because we are interested in application run time, the speed of decompression is of much greater importance.

Wynn Article, 3.2



3.8. Test sequence

We created a test sequence which would approximate the sequence of events followed by a typical user. The first step of the test sequence is to power up the computer, which will load the operating system. Then a series of folders and applications, including word-processor, calendar application and realtime video, are opened. The applications are executed, then the folders and applications are closed.

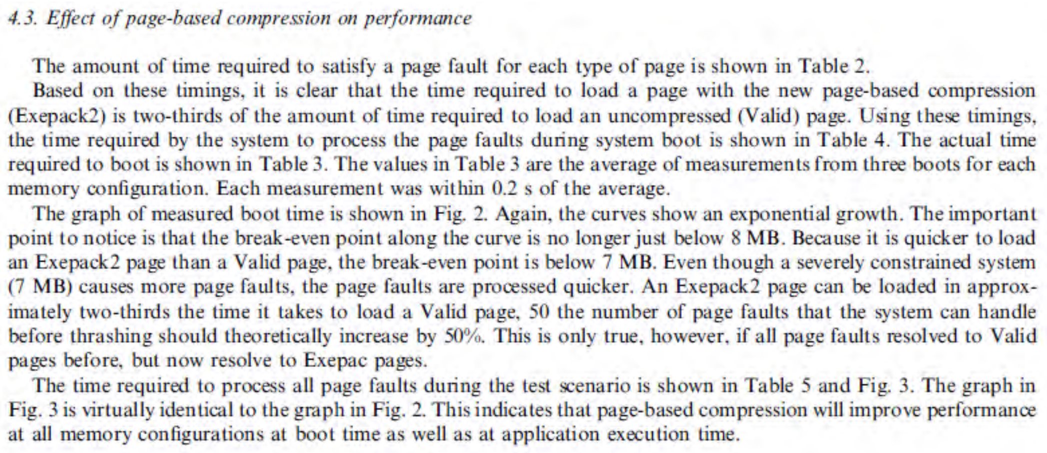
This test sequence is an approximation of the steps followed by a user during a session at the computer. First, the user must power on the computer. Then the user will start and execute a number of applications. In order to start an application, the folder containing the application must be opened. Finally, when the user is finished with an application and/or folder, it will be closed. At the end of the session, all objects which have been opened will be closed before the system is powered off.

The sequence of events used to collect data is listed below. The system was allowed to quiesce completely between each step before the data ensuring all paging activity has finished.

Test Sequence

boot	Power on the computer
1	Start IBM Works folder
2	Open IBM Works Application
3	Create new document
4	Close IMB Works application

Wynn Article, 3.8



4.3. Effect of page-based compression on performance

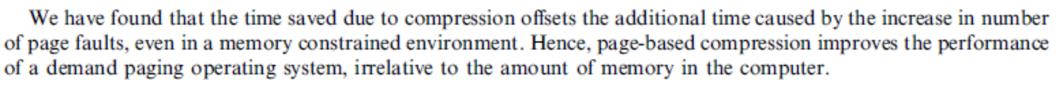
The amount of time required to satisfy a page fault for each type of page is shown in Table 2.

Based on these timings, it is clear that the time required to load a page with the new page-based compression (Exepack2) is two-thirds of the amount of time required to load an uncompressed (Valid) page. Using these timings, the time required by the system to process the page faults during system boot is shown in Table 4. The actual time required to boot is shown in Table 3. The values in Table 3 are the average of measurements from three boots for each memory configuration. Each measurement was within 0.2 s of the average.

The graph of measured boot time is shown in Fig. 2. Again, the curves show an exponential growth. The important point to notice is that the break-even point along the curve is no longer just below 8 MB. Because it is quicker to load an Exepack2 page than a Valid page, the break-even point is below 7 MB. Even though a severely constrained system (7 MB) causes more page faults, the page faults are processed quicker. An Exepack2 page can be loaded in approximately two-thirds the time it takes to load a Valid page, 50 the number of page faults that the system can handle before thrashing should theoretically increase by 50%. This is only true, however, if all page faults resolved to Valid pages before, but now resolve to Exepac pages.

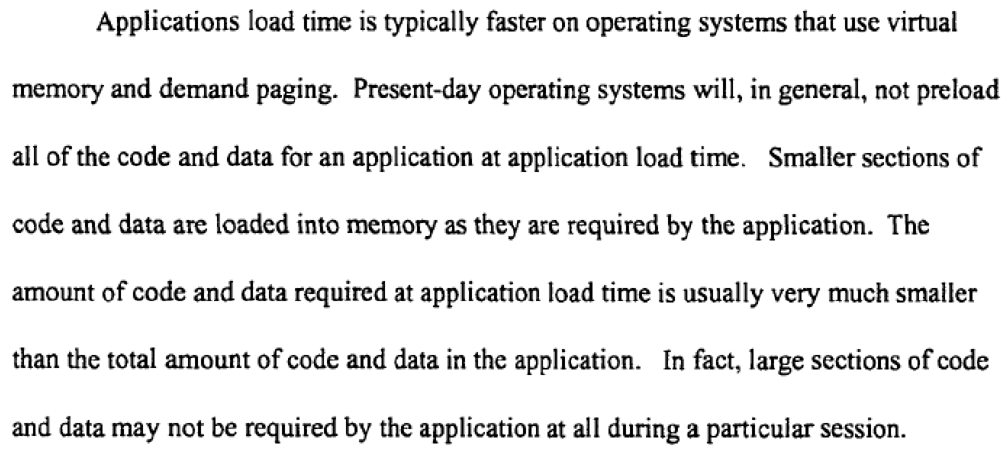
The time required to process all page faults during the test scenario is shown in Table 5 and Fig. 3. The graph in Fig. 3 is virtually identical to the graph in Fig. 2. This indicates that page-based compression will improve performance at all memory configurations at boot time as well as at application execution time.

Wynn Article, 4.3



We have found that the time saved due to compression offsets the additional time caused by the increase in number of page faults, even in a memory constrained environment. Hence, page-based compression improves the performance of a demand paging operating system, irrelative to the amount of memory in the computer.

Wynn Article, 1.2



Applications load time is typically faster on operating systems that use virtual memory and demand paging. Present-day operating systems will, in general, not preload all of the code and data for an application at application load time. Smaller sections of code and data are loaded into memory as they are required by the application. The amount of code and data required at application load time is usually very much smaller than the total amount of code and data in the application. In fact, large sections of code and data may not be required by the application at all during a particular session.

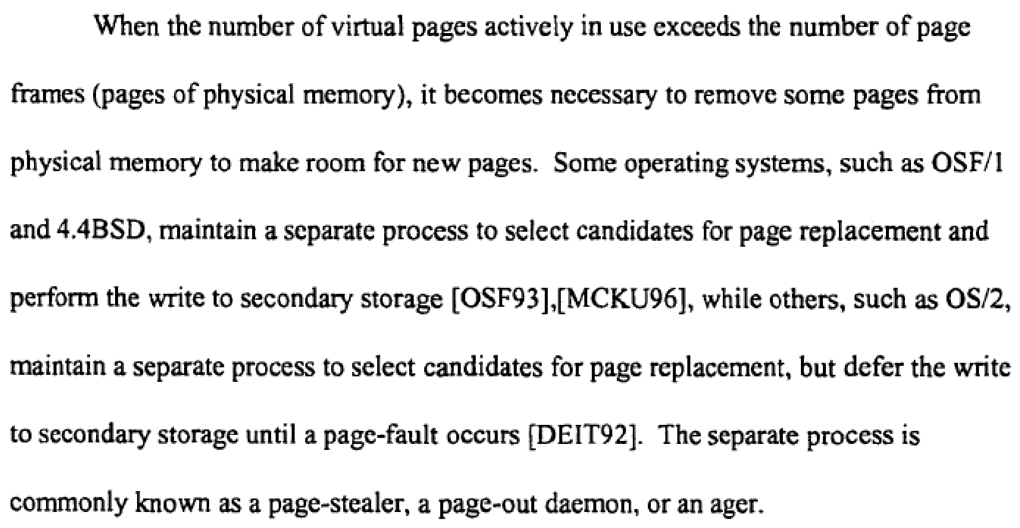
Wynn Thesis, 2.1

5. The method of claim 1, wherein said preloading is performed by a data storage controller connected to said boot device.	Wynn, as evidenced by the example citations below, discloses "said preloading is performed by a data storage controller connected to said boot device."

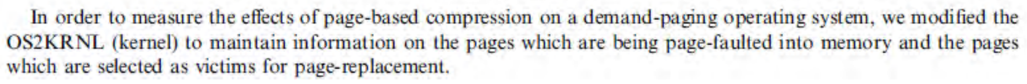
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said preloading is performed by a data storage controller connected to said boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Wynn discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.


When the number of virtual pages actively in use exceeds the number of page frames (pages of physical memory), it becomes necessary to remove some pages from physical memory to make room for new pages. Some operating systems, such as OSF/1 and 4.4BSD, maintain a separate process to select candidates for page replacement and perform the write to secondary storage [OSF93],[MCKU96], while others, such as OS/2, maintain a separate process to select candidates for page replacement, but defer the write to secondary storage until a page-fault occurs [DEIT92]. The separate process is commonly known as a page-stealer, a page-out daemon, or an ager.

Wynn Thesis, 2.3


In order to measure the effects of page-based compression on a demand-paging operating system, we modified the OS2KRNL (kernel) to maintain information on the pages which are being page-faulted into memory and the pages which are selected as victims for page-replacement.

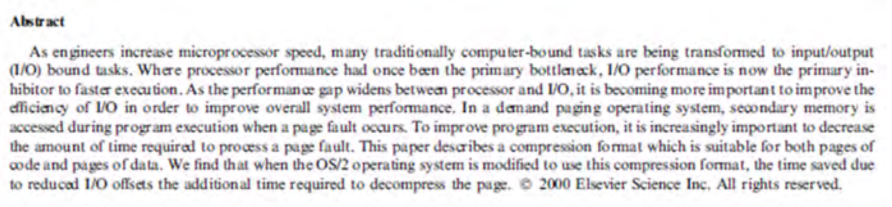
Wynn Article, 3.6

6. The method of claim 1, further comprising updating the list of boot data.	Wynn, as evidenced by the example citations below, discloses "updating the list of boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Wynn discloses this limitation:

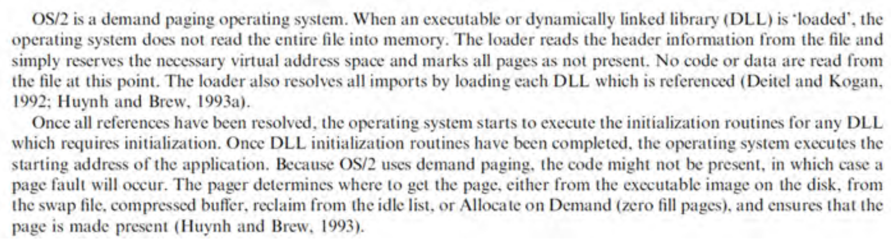
See Claims 1.1, 1.3, and 1.4 above.



Abstract

As engineers increase microprocessor speed, many traditionally computer-bound tasks are being transformed to input/output (I/O) bound tasks. Where processor performance had once been the primary bottleneck, I/O performance is now the primary inhibitor to faster execution. As the performance gap widens between processor and I/O, it is becoming more important to improve the efficiency of I/O in order to improve overall system performance. In a demand paging operating system, secondary memory is accessed during program execution when a page fault occurs. To improve program execution, it is increasingly important to decrease the amount of time required to process a page fault. This paper describes a compression format which is suitable for both pages of code and pages of data. We find that when the OS/2 operating system is modified to use this compression format, the time saved due to reduced I/O offsets the additional time required to decompress the page. © 2000 Elsevier Science Inc. All rights reserved.

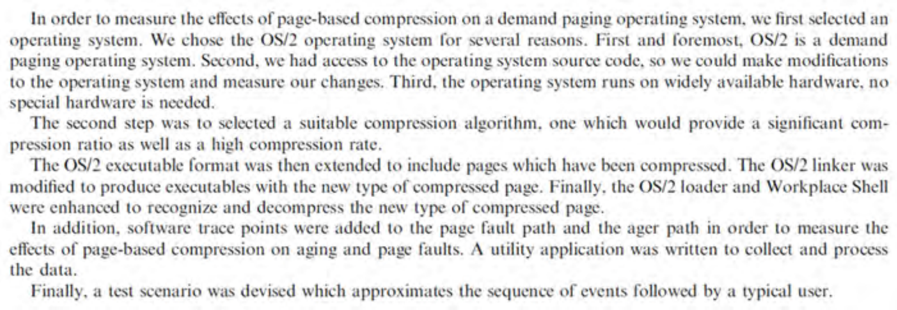
Wynn Article, Abstract



OS/2 is a demand paging operating system. When an executable or dynamically linked library (DLL) is 'loaded', the operating system does not read the entire file into memory. The loader reads the header information from the file and simply reserves the necessary virtual address space and marks all pages as not present. No code or data are read from the file at this point. The loader also resolves all imports by loading each DLL which is referenced (Deitel and Kogan, 1992; Huynh and Brew, 1993a).

Once all references have been resolved, the operating system starts to execute the initialization routines for any DLL which requires initialization. Once DLL initialization routines have been completed, the operating system executes the starting address of the application. Because OS/2 uses demand paging, the code might not be present, in which case a page fault will occur. The pager determines where to get the page, either from the executable image on the disk, from the swap file, compressed buffer, reclaim from the idle list, or Allocate on Demand (zero fill pages), and ensures that the page is made present (Huynh and Brew, 1993).

Wynn Article, 2.2



In order to measure the effects of page-based compression on a demand paging operating system, we first selected an operating system. We chose the OS/2 operating system for several reasons. First and foremost, OS/2 is a demand paging operating system. Second, we had access to the operating system source code, so we could make modifications to the operating system and measure our changes. Third, the operating system runs on widely available hardware, no special hardware is needed.

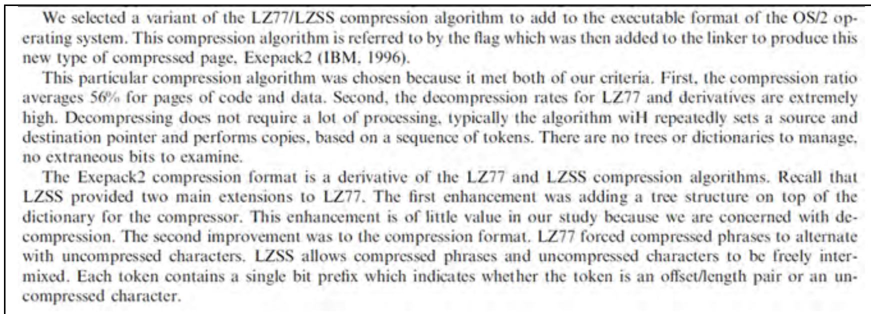
The second step was to selected a suitable compression algorithm, one which would provide a significant compression ratio as well as a high compression rate.

The OS/2 executable format was then extended to include pages which have been compressed. The OS/2 linker was modified to produce executables with the new type of compressed page. Finally, the OS/2 loader and Workplace Shell were enhanced to recognize and decompress the new type of compressed page.

In addition, software trace points were added to the page fault path and the ager path in order to measure the effects of page-based compression on aging and page faults. A utility application was written to collect and process the data.

Finally, a test scenario was devised which approximates the sequence of events followed by a typical user.

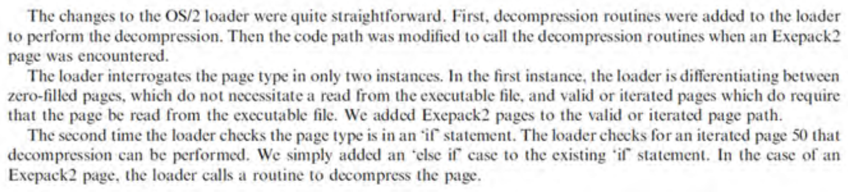
Wynn Article, 3.1


We selected a variant of the LZ77/LZSS compression algorithm to add to the executable format of the OS/2 operating system. This compression algorithm is referred to by the flag which was then added to the linker to produce this new type of compressed page, Exepack2 (IBM, 1996).

This particular compression algorithm was chosen because it met both of our criteria. First, the compression ratio averages 56% for pages of code and data. Second, the decompression rates for LZ77 and derivatives are extremely high. Decompressing does not require a lot of processing, typically the algorithm wiH repeatedly sets a source and destination pointer and performs copies, based on a sequence of tokens. There are no trees or dictionaries to manage, no extraneous bits to examine.

The Exepack2 compression format is a derivative of the LZ77 and LZSS compression algorithms. Recall that LZSS provided two main extensions to LZ77. The first enhancement was adding a tree structure on top of the dictionary for the compressor. This enhancement is of little value in our study because we are concerned with decompression. The second improvement was to the compression format. LZ77 forced compressed phrases to alternate with uncompressed characters. LZSS allows compressed phrases and uncompressed characters to be freely intermixed. Each token contains a single bit prefix which indicates whether the token is an offset/length pair or an uncompressed character.

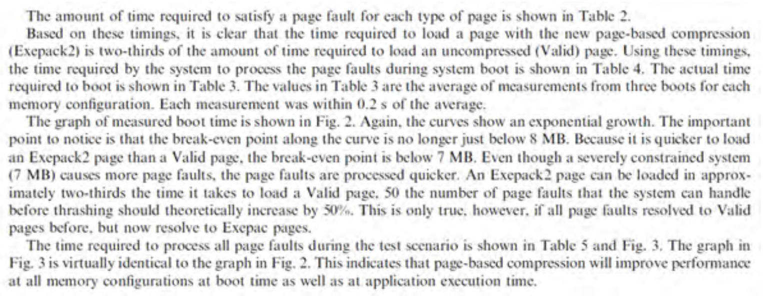
Wynn Article, 3.2


The changes to the OS/2 loader were quite straightforward. First, decompression routines were added to the loader to perform the decompression. Then the code path was modified to call the decompression routines when an Exepack2 page was encountered.

The loader interrogates the page type in only two instances. In the first instance, the loader is differentiating between zero-filled pages, which do not necessitate a read from the executable file, and valid or iterated pages which do require that the page be read from the executable file. We added Exepack2 pages to the valid or iterated page path.

The second time the loader checks the page type is in an 'if' statement. The loader checks for an iterated page 50 that decompression can be performed. We simply added an 'else if' case to the existing 'if' statement. In the case of an Exepack2 page, the loader calls a routine to decompress the page.

Wynn Article, 3.4


The amount of time required to satisfy a page fault for each type of page is shown in Table 2.

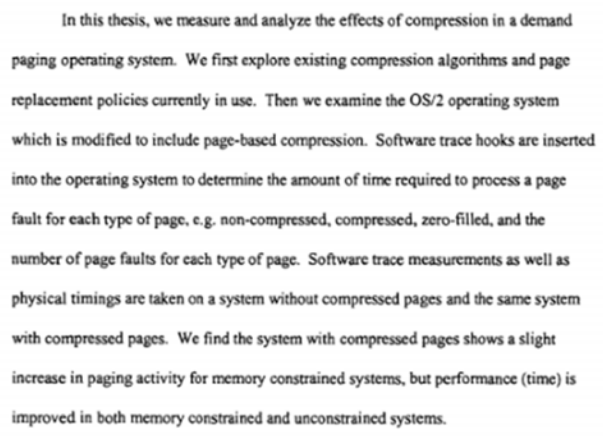
Based on these timings, it is clear that the time required to load a page with the new page-based compression (Exepack2) is two-thirds of the amount of time required to load an uncompressed (Valid) page. Using these timings, the time required by the system to process the page faults during system boot is shown in Table 4. The actual time required to boot is shown in Table 3. The values in Table 3 are the average of measurements from three boots for each memory configuration. Each measurement was within 0.2 s of the average.

The graph of measured boot time is shown in Fig. 2. Again, the curves show an exponential growth. The important point to notice is that the break-even point along the curve is no longer just below 8 MB. Because it is quicker to load an Exepack2 page than a Valid page, the break-even point is below 7 MB. Even though a severely constrained system (7 MB) causes more page faults, the page faults are processed quicker. An Exepack2 page can be loaded in approximately two-thirds the time it takes to load a Valid page, 50 the number of page faults that the system can handle before thrashing should theoretically increase by 50%. This is only true, however, if all page faults resolved to Valid pages before, but now resolve to Exepac pages.

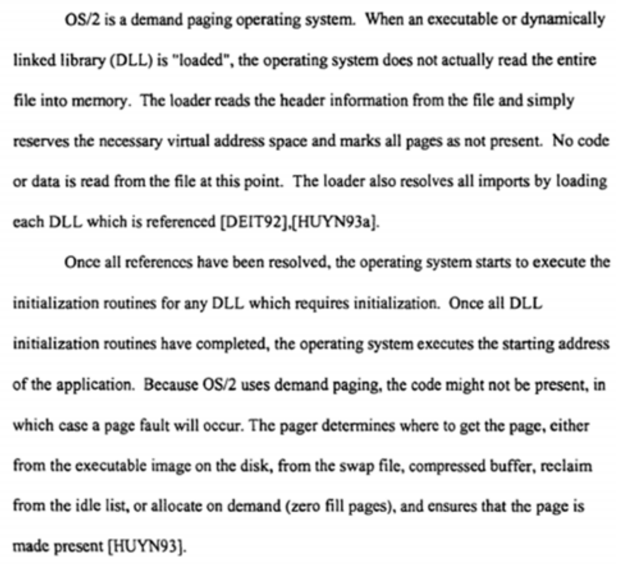
The time required to process all page faults during the test scenario is shown in Table 5 and Fig. 3. The graph in Fig. 3 is virtually identical to the graph in Fig. 2. This indicates that page-based compression will improve performance at all memory configurations at boot time as well as at application execution time.

Wynn Article, 4.3

See also, Wynn Article, Table 3 and Fig. 2


In this thesis, we measure and analyze the effects of compression in a demand paging operating system. We first explore existing compression algorithms and page replacement policies currently in use. Then we examine the OS/2 operating system which is modified to include page-based compression. Software trace hooks are inserted into the operating system to determine the amount of time required to process a page fault for each type of page, e.g. non-compressed, compressed, zero-filled, and the number of page faults for each type of page. Software trace measurements as well as physical timings are taken on a system without compressed pages and the same system with compressed pages. We find the system with compressed pages shows a slight increase in paging activity for memory constrained systems, but performance (time) is improved in both memory constrained and unconstrained systems.

Wynn Thesis, Abstract


OS/2 is a demand paging operating system. When an executable or dynamically linked library (DLL) is "loaded", the operating system does not actually read the entire file into memory. The loader reads the header information from the file and simply reserves the necessary virtual address space and marks all pages as not present. No code or data is read from the file at this point. The loader also resolves all imports by loading each DLL which is referenced [DEIT92],[HUYN93a].

Once all references have been resolved, the operating system starts to execute the initialization routines for any DLL which requires initialization. Once all DLL initialization routines have completed, the operating system executes the starting address of the application. Because OS/2 uses demand paging, the code might not be present, in which case a page fault will occur. The pager determines where to get the page, either from the executable image on the disk, from the swap file, compressed buffer, reclaim from the idle list, or allocate on demand (zero fill pages), and ensures that the page is made present [HUYN93].

Wynn Thesis, 4.2

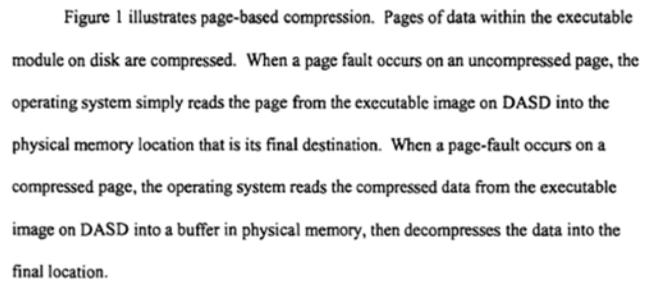
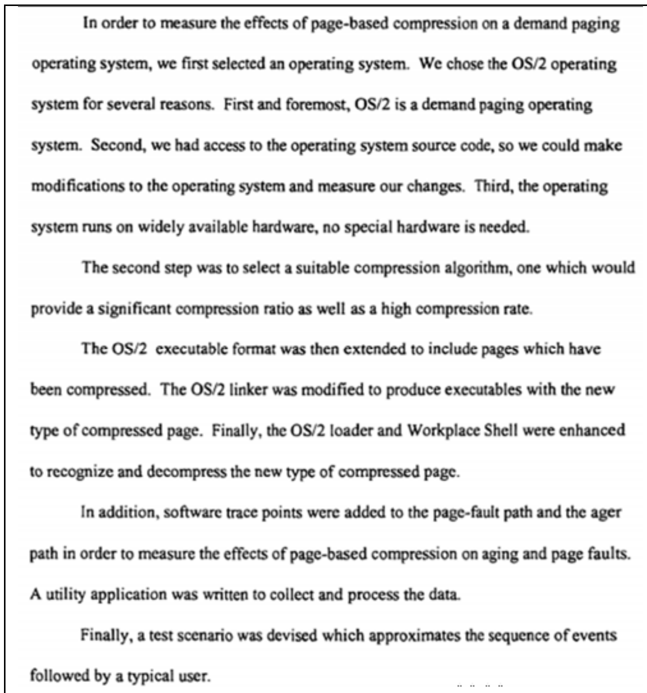


Figure 1 illustrates page-based compression. Pages of data within the executable module on disk are compressed. When a page fault occurs on an uncompressed page, the operating system simply reads the page from the executable image on DASD into the physical memory location that is its final destination. When a page-fault occurs on a compressed page, the operating system reads the compressed data from the executable image on DASD into a buffer in physical memory, then decompresses the data into the final location.


In order to measure the effects of page-based compression on a demand paging operating system, we first selected an operating system. We chose the OS/2 operating system for several reasons. First and foremost, OS/2 is a demand paging operating system. Second, we had access to the operating system source code, so we could make modifications to the operating system and measure our changes. Third, the operating system runs on widely available hardware, no special hardware is needed.

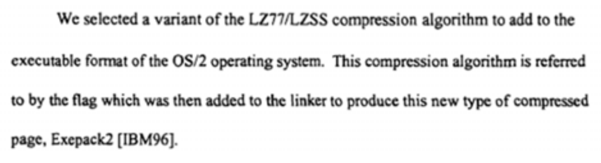
The second step was to select a suitable compression algorithm, one which would provide a significant compression ratio as well as a high compression rate.

The OS/2 executable format was then extended to include pages which have been compressed. The OS/2 linker was modified to produce executables with the new type of compressed page. Finally, the OS/2 loader and Workplace Shell were enhanced to recognize and decompress the new type of compressed page.

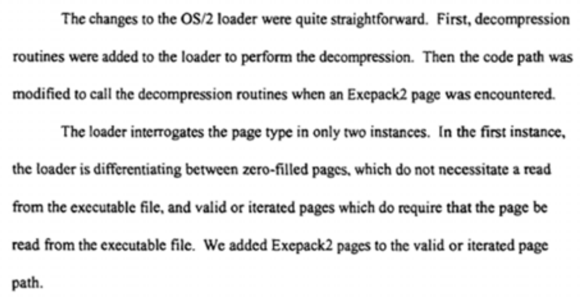
In addition, software trace points were added to the page-fault path and the ager path in order to measure the effects of page-based compression on aging and page faults. A utility application was written to collect and process the data.

Finally, a test scenario was devised which approximates the sequence of events followed by a typical user.

Wynn Thesis, 5.1

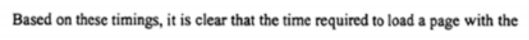

We selected a variant of the LZ77/LZSS compression algorithm to add to the executable format of the OS/2 operating system. This compression algorithm is referred to by the flag which was then added to the linker to produce this new type of compressed page, Exepack2 [IBM96].

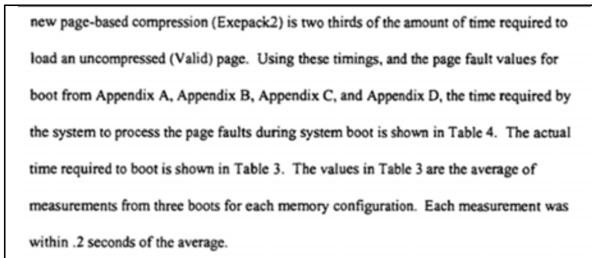
Wynn Thesis, 5.2


The changes to the OS/2 loader were quite straightforward. First, decompression routines were added to the loader to perform the decompression. Then the code path was modified to call the decompression routines when an Exepack2 page was encountered.

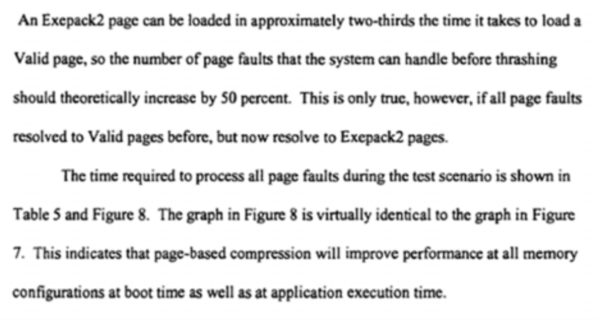
The loader interrogates the page type in only two instances. In the first instance, the loader is differentiating between zero-filled pages, which do not necessitate a read from the executable file, and valid or iterated pages which do require that the page be read from the executable file. We added Exepack2 pages to the valid or iterated page path.

Wynn Thesis, 5.5


Based on these timings, it is clear that the time required to load a page with the

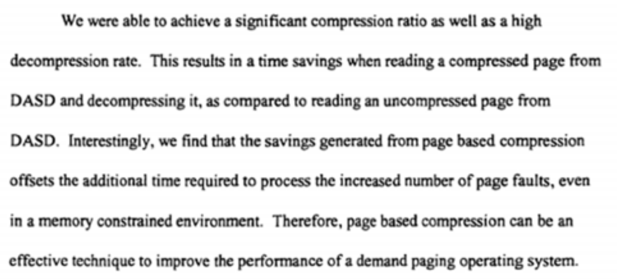

new page-based compression (Exepack2) is two thirds of the amount of time required to load an uncompressed (Valid) page. Using these timings, and the page fault values for boot from Appendix A, Appendix B, Appendix C, and Appendix D, the time required by the system to process the page faults during system boot is shown in Table 4. The actual time required to boot is shown in Table 3. The values in Table 3 are the average of measurements from three boots for each memory configuration. Each measurement was within .2 seconds of the average.

Wynn Thesis, 6.3


An Exepack2 page can be loaded in approximately two-thirds the time it takes to load a Valid page, so the number of page faults that the system can handle before thrashing should theoretically increase by 50 percent. This is only true, however, if all page faults resolved to Valid pages before, but now resolve to Exepack2 pages.

The time required to process all page faults during the test scenario is shown in Table 5 and Figure 8. The graph in Figure 8 is virtually identical to the graph in Figure 7. This indicates that page-based compression will improve performance at all memory configurations at boot time as well as at application execution time.

Wynn Thesis, 6.3


We were able to achieve a significant compression ratio as well as a high decompression rate. This results in a time savings when reading a compressed page from DASD and decompressing it, as compared to reading an uncompressed page from DASD. Interestingly, we find that the savings generated from page based compression offsets the additional time required to process the increased number of page faults, even in a memory constrained environment. Therefore, page based compression can be an effective technique to improve the performance of a demand paging operating system.

Wynn Thesis, 7.2

See also Wynn Thesis, Table 2-4, Fig. 7

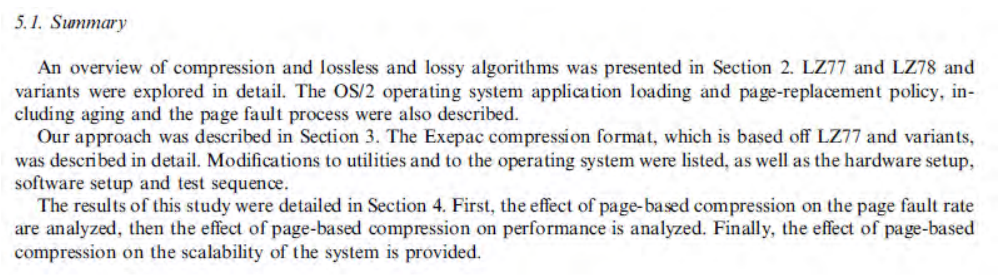
Appendix B37
Invalidity of U.S. Patent 8,090,936 based on Wynn

8. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form.	Wynn, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Wynn discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.



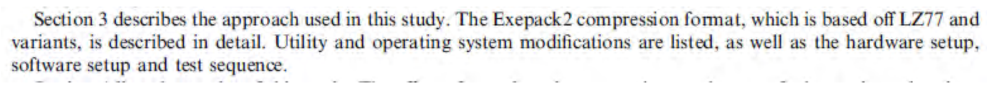
5.1. Summary

An overview of compression and lossless and lossy algorithms was presented in Section 2. LZ77 and LZ78 and variants were explored in detail. The OS/2 operating system application loading and page-replacement policy, including aging and the page fault process were also described.

Our approach was described in Section 3. The Exepac compression format, which is based off LZ77 and variants, was described in detail. Modifications to utilities and to the operating system were listed, as well as the hardware setup, software setup and test sequence.

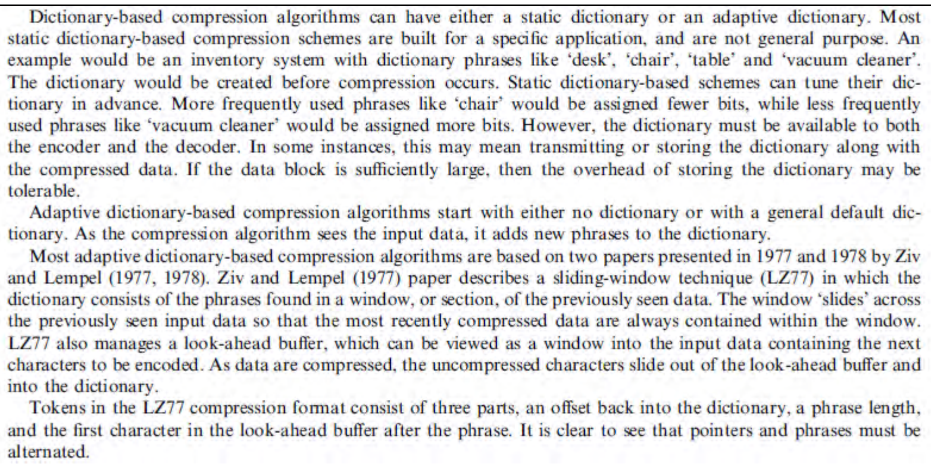
The results of this study were detailed in Section 4. First, the effect of page-based compression on the page fault rate are analyzed, then the effect of page-based compression on performance is analyzed. Finally, the effect of page-based compression on the scalability of the system is provided.

Wynn Article, 5.1



Section 3 describes the approach used in this study. The Exepack2 compression format, which is based off LZ77 and variants, is described in detail. Utility and operating system modifications are listed, as well as the hardware setup, software setup and test sequence.

Wynn Article, 1.2



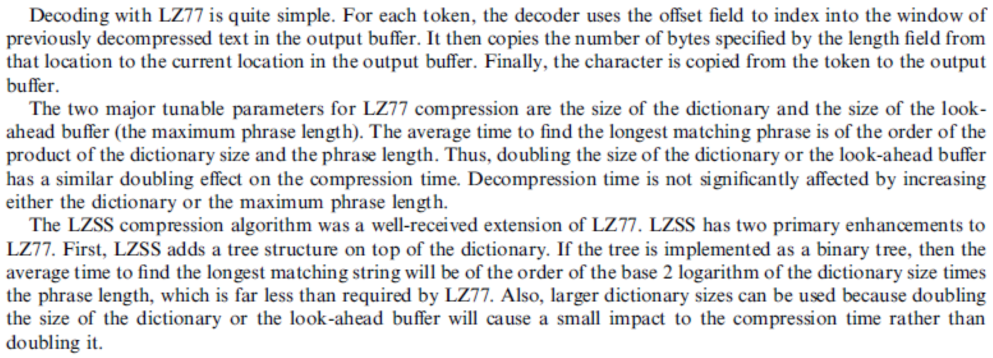
Dictionary-based compression algorithms can have either a static dictionary or an adaptive dictionary. Most static dictionary-based compression schemes are built for a specific application, and are not general purpose. An example would be an inventory system with dictionary phrases like 'desk', 'chair', 'table' and 'vacuum cleaner'. The dictionary would be created before compression occurs. Static dictionary-based schemes can tune their dictionary in advance. More frequently used phrases like 'chair' would be assigned fewer bits, while less frequently used phrases like 'vacuum cleaner' would be assigned more bits. However, the dictionary must be available to both the encoder and the decoder. In some instances, this may mean transmitting or storing the dictionary along with the compressed data. If the data block is sufficiently large, then the overhead of storing the dictionary may be tolerable.

Adaptive dictionary-based compression algorithms start with either no dictionary or with a general default dictionary. As the compression algorithm sees the input data, it adds new phrases to the dictionary.

Most adaptive dictionary-based compression algorithms are based on two papers presented in 1977 and 1978 by Ziv and Lempel (1977, 1978). Ziv and Lempel (1977) paper describes a sliding-window technique (LZ77) in which the dictionary consists of the phrases found in a window, or section, of the previously seen data. The window 'slides' across the previously seen input data so that the most recently compressed data are always contained within the window. LZ77 also manages a look-ahead buffer, which can be viewed as a window into the input data containing the next characters to be encoded. As data are compressed, the uncompressed characters slide out of the look-ahead buffer and into the dictionary.

Tokens in the LZ77 compression format consist of three parts, an offset back into the dictionary, a phrase length, and the first character in the look-ahead buffer after the phrase. It is clear to see that pointers and phrases must be alternated.

Wynn Article, 2.1

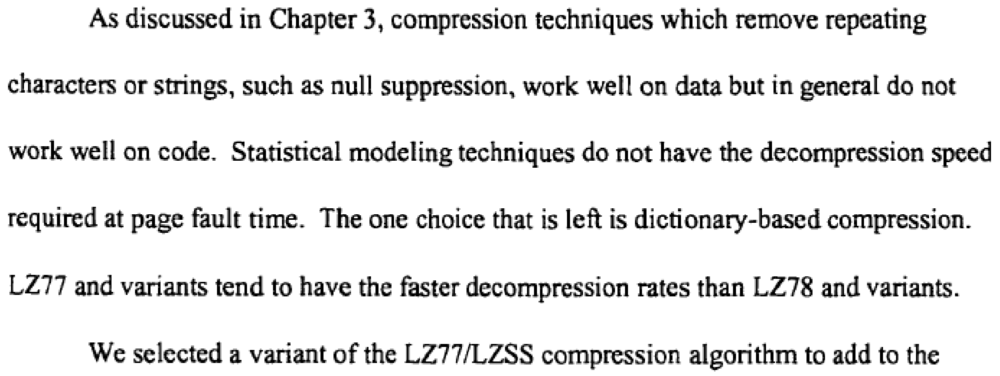


Decoding with LZ77 is quite simple. For each token, the decoder uses the offset field to index into the window of previously decompressed text in the output buffer. It then copies the number of bytes specified by the length field from that location to the current location in the output buffer. Finally, the character is copied from the token to the output buffer.

The two major tunable parameters for LZ77 compression are the size of the dictionary and the size of the look-ahead buffer (the maximum phrase length). The average time to find the longest matching phrase is of the order of the product of the dictionary size and the phrase length. Thus, doubling the size of the dictionary or the look-ahead buffer has a similar doubling effect on the compression time. Decompression time is not significantly affected by increasing either the dictionary or the maximum phrase length.

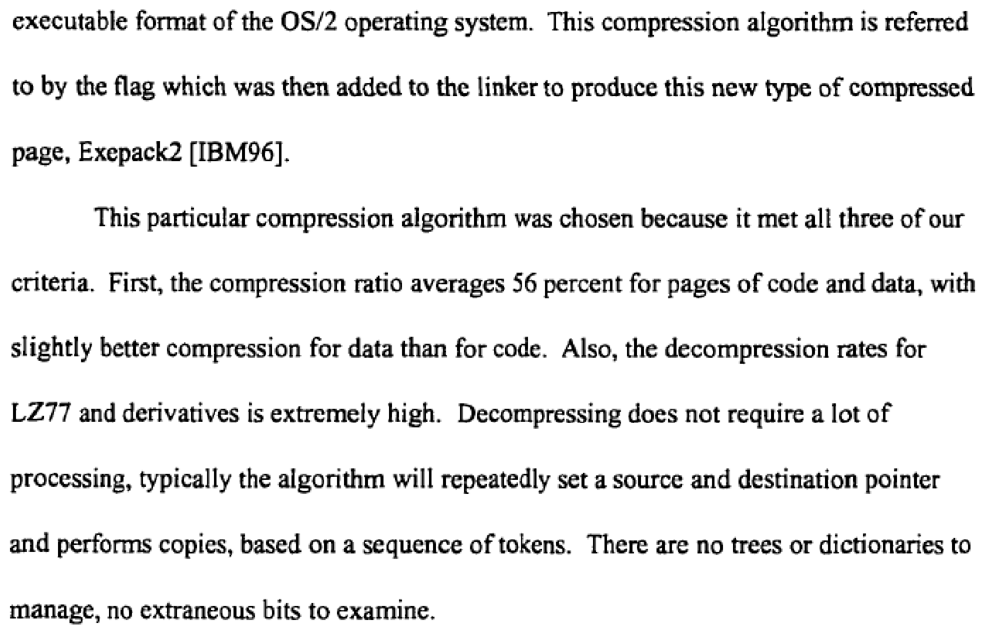
The LZSS compression algorithm was a well-received extension of LZ77. LZSS has two primary enhancements to LZ77. First, LZSS adds a tree structure on top of the dictionary. If the tree is implemented as a binary tree, then the average time to find the longest matching string will be of the order of the base 2 logarithm of the dictionary size times the phrase length, which is far less than required by LZ77. Also, larger dictionary sizes can be used because doubling the size of the dictionary or the look-ahead buffer will cause a small impact to the compression time rather than doubling it.

Wynn Article, 2.1



As discussed in Chapter 3, compression techniques which remove repeating characters or strings, such as null suppression, work well on data but in general do not work well on code. Statistical modeling techniques do not have the decompression speed required at page fault time. The one choice that is left is dictionary-based compression. LZ77 and variants tend to have the faster decompression rates than LZ78 and variants.

We selected a variant of the LZ77/LZSS compression algorithm to add to the


executable format of the OS/2 operating system. This compression algorithm is referred to by the flag which was then added to the linker to produce this new type of compressed page, Exepack2 [IBM96].

This particular compression algorithm was chosen because it met all three of our criteria. First, the compression ratio averages 56 percent for pages of code and data, with slightly better compression for data than for code. Also, the decompression rates for LZ77 and derivatives is extremely high. Decompressing does not require a lot of processing, typically the algorithm will repeatedly set a source and destination pointer and performs copies, based on a sequence of tokens. There are no trees or dictionaries to manage, no extraneous bits to examine.

Wynn Thesis, 5.2

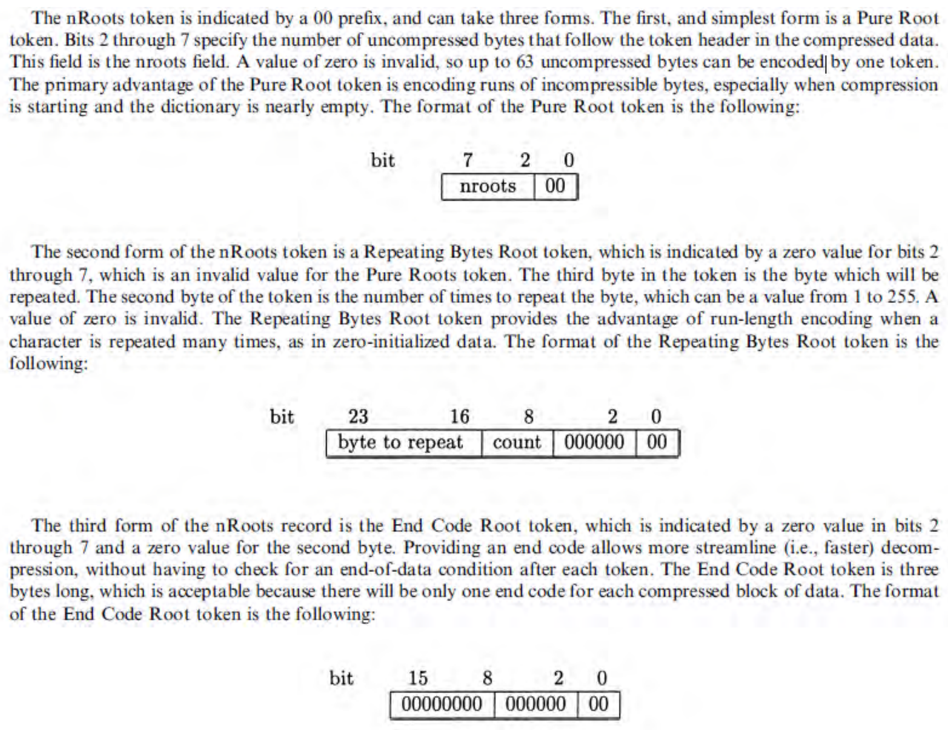
Appendix B37
Invalidity of U.S. Patent 8,090,936 based on Wynn

9. The method of claim 1, wherein a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form.	Wynn, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Wynn discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.



The nRoots token is indicated by a 00 prefix, and can take three forms. The first, and simplest form is a Pure Root token. Bits 2 through 7 specify the number of uncompressed bytes that follow the token header in the compressed data. This field is the nroots field. A value of zero is invalid, so up to 63 uncompressed bytes can be encoded by one token. The primary advantage of the Pure Root token is encoding runs of incompressible bytes, especially when compression is starting and the dictionary is nearly empty. The format of the Pure Root token is the following:

```
bit  7       2  0
    | nroots | 00 |
```

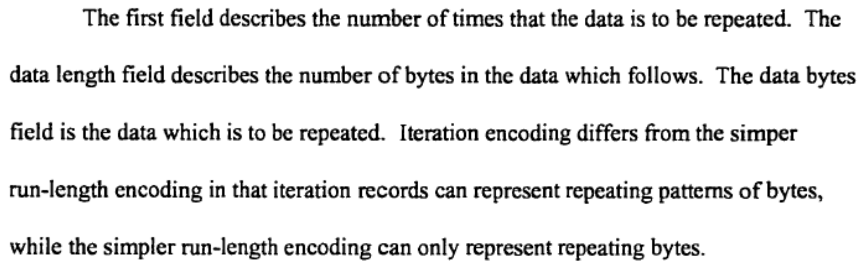
The second form of the nRoots token is a Repeating Bytes Root token, which is indicated by a zero value for bits 2 through 7, which is an invalid value for the Pure Roots token. The third byte in the token is the byte which will be repeated. The second byte of the token is the number of times to repeat the byte, which can be a value from 1 to 255. A value of zero is invalid. The Repeating Bytes Root token provides the advantage of run-length encoding when a character is repeated many times, as in zero-initialized data. The format of the Repeating Bytes Root token is the following:

```
bit  23             16      8        2  0
    | byte to repeat | count | 000000 | 00 |
```

The third form of the nRoots record is the End Code Root token, which is indicated by a zero value in bits 2 through 7 and a zero value for the second byte. Providing an end code allows more streamline (i.e., faster) decompression, without having to check for an end-of-data condition after each token. The End Code Root token is three bytes long, which is acceptable because there will be only one end code for each compressed block of data. The format of the End Code Root token is the following:

```
bit  15         8        2  0
    | 00000000 | 000000 | 00 |
```

Wynn Article, 3.2


The first field describes the number of times that the data is to be repeated. The data length field describes the number of bytes in the data which follows. The data bytes field is the data which is to be repeated. Iteration encoding differs from the simper run-length encoding in that iteration records can represent repeating patterns of bytes, while the simpler run-length encoding can only represent repeating bytes.

Wynn Thesis, 5.3

11.1. a processor;	Wynn, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Wynn discloses this limitation: *See* Claim 1.2 above.	

Appendix B37
Invalidity of U.S. Patent 8,090,936 based on Wynn

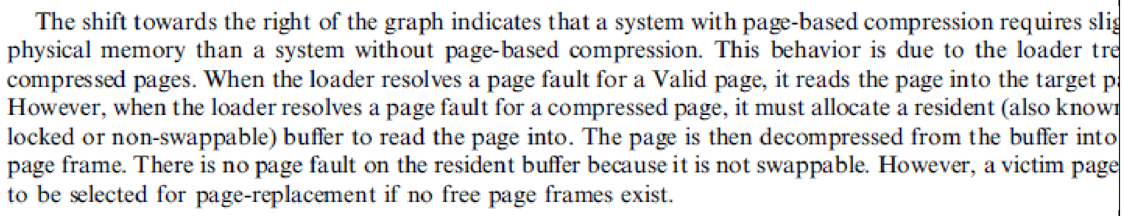
11.2. a memory; and	Wynn, as evidenced by the example citations below, discloses “a memory.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Wynn discloses this limitation: *See* Claims 1.3, and 1.4 above.	

11.3.1 a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device,	Wynn, as evidenced by the example citations below, discloses "a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

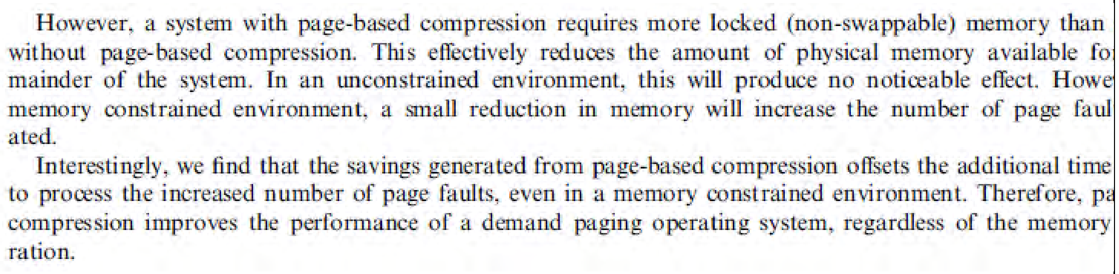
Wynn discloses this limitation:

See Claims 1.1, 1.3, 1.4 above.



The shift towards the right of the graph indicates that a system with page-based compression requires slig physical memory than a system without page-based compression. This behavior is due to the loader tre compressed pages. When the loader resolves a page fault for a Valid page, it reads the page into the target pa However, when the loader resolves a page fault for a compressed page, it must allocate a resident (also knowi locked or non-swappable) buffer to read the page into. The page is then decompressed from the buffer into page frame. There is no page fault on the resident buffer because it is not swappable. However, a victim page to be selected for page-replacement if no free page frames exist.

Wynn Article, 4.2



However, a system with page-based compression requires more locked (non-swappable) memory than without page-based compression. This effectively reduces the amount of physical memory available fo mainder of the system. In an unconstrained environment, this will produce no noticeable effect. Howe memory constrained environment, a small reduction in memory will increase the number of page faul ated.

Interestingly, we find that the savings generated from page-based compression offsets the additional time to process the increased number of page faults, even in a memory constrained environment. Therefore, pa compression improves the performance of a demand paging operating system, regardless of the memory ration.

Wynn Article, 5.2

11.3.2 said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system; and,	Wynn, as evidenced by the example citations below, discloses "said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Wynn discloses this limitation: *See* Claims 1.3, 1.4, and 1.5 above.	

11.4 a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data.	Wynn, as evidenced by the example citations below, discloses “a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Wynn discloses this limitation: *See* Claims 1.1 and 1.5 above.	

12. The system of claim 11, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program.	Wynn, as evidenced by the example citations below, discloses “said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Wynn discloses this limitation: *See* Claims 1.1, 3, and 11.3.1 above.	

13. The system of claim 11, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system.	Wynn, as evidenced by the example citations below, discloses "wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Wynn discloses this limitation: *See* Claims 1.1, 1.3, 3, 5 and 11.3.1 and 11.3.2 above.	

15. The system of claim 11, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form.	Wynn, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Wynn discloses this limitation: *See* Claims 1, 8, and 11 above.	

16. The system of claim 11, wherein a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form.	Wynn, as evidenced by the example citations below, discloses “a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Wynn discloses this limitation: *See* Claims 1, 9, and 11 above.	

Appendix B38
Invalidity of U.S. Patent 8,090,936 based on Linux Kernel

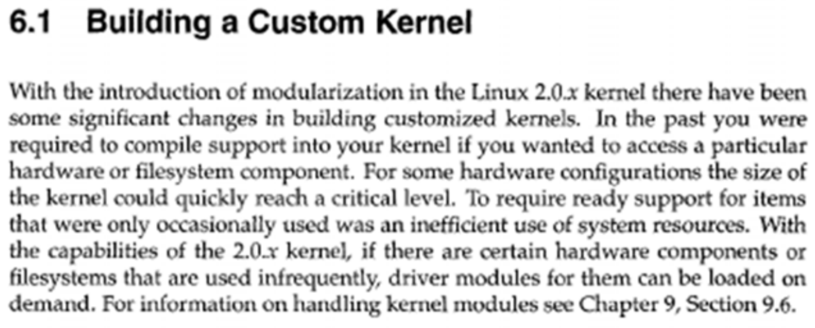
Red Hat Linux 5.0, The Official Red Hat Linux Installation Guide (1995) ("Linux Redhat") and M. Beck, et. al, "Linux Kernel Internals" Addison Wesley Longman (1996) ("Beck") (collectively, "Linux Kernel"), alone or in combination, invalidate claims 1-6, 8-9, 11-13, and 15-16 of United States Patent No. 8,090,936 ("the '936 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '936 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

1.1 maintaining a list of boot data used for booting a computer system, wherein at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device;	Linux Kernel, as evidenced by the exemplary citations below, discloses "maintaining a list of boot data used for booting a computer system, wherein at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, maintaining a list of boot data used for booting a computer system, wherein at least a portion of said boot data is compressed by a data compression engine to provide said at least a portion of said boot data in compressed form, and stored in compressed form on a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

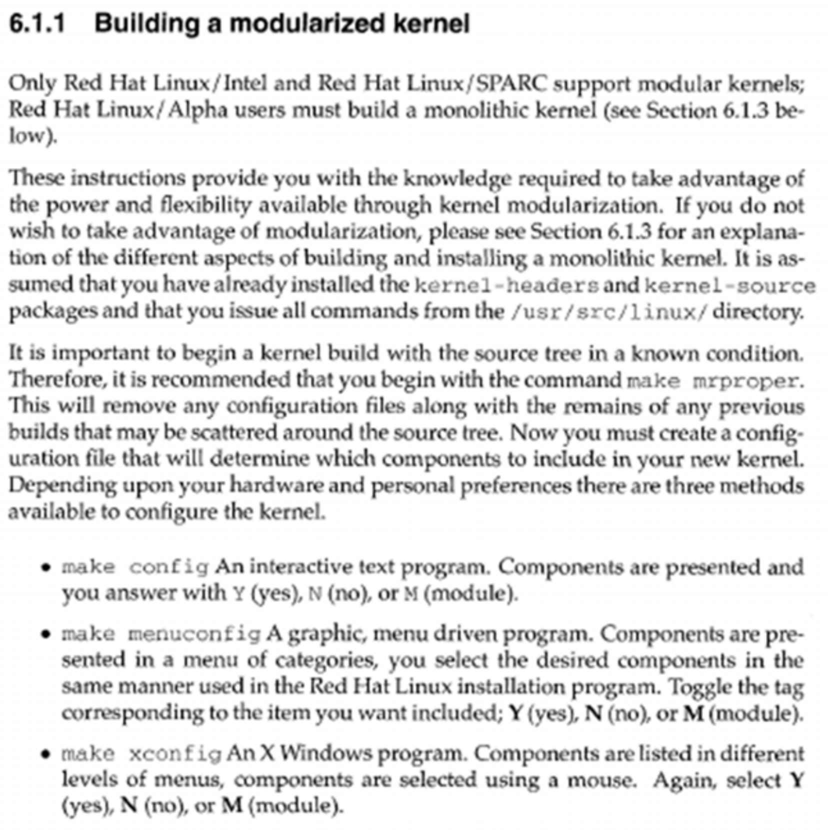
Linux Kernel discloses this limitation:



6.1 Building a Custom Kernel

With the introduction of modularization in the Linux 2.0.*x* kernel there have been some significant changes in building customized kernels. In the past you were required to compile support into your kernel if you wanted to access a particular hardware or filesystem component. For some hardware configurations the size of the kernel could quickly reach a critical level. To require ready support for items that were only occasionally used was an inefficient use of system resources. With the capabilities of the 2.0.*x* kernel, if there are certain hardware components or filesystems that are used infrequently, driver modules for them can be loaded on demand. For information on handling kernel modules see Chapter 9, Section 9.6.

Linux Redhat, at 6.1



6.1.1 Building a modularized kernel

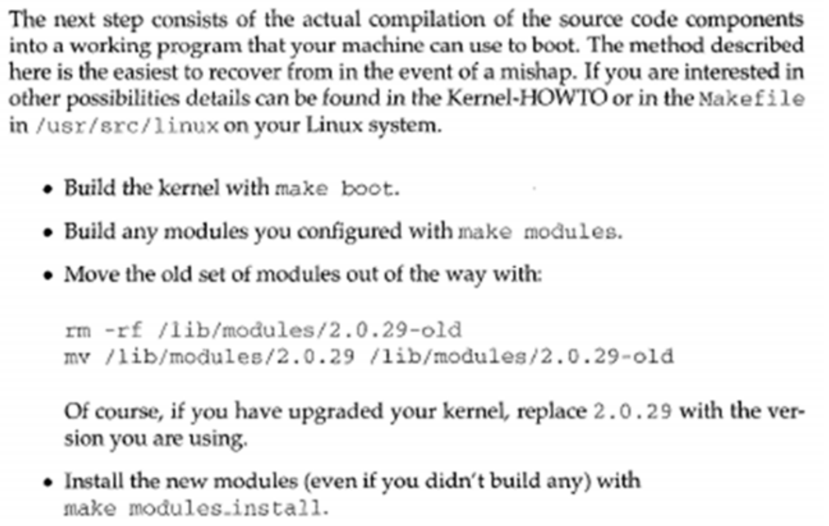
Only Red Hat Linux/Intel and Red Hat Linux/SPARC support modular kernels; Red Hat Linux/Alpha users must build a monolithic kernel (see Section 6.1.3 below).

These instructions provide you with the knowledge required to take advantage of the power and flexibility available through kernel modularization. If you do not wish to take advantage of modularization, please see Section 6.1.3 for an explanation of the different aspects of building and installing a monolithic kernel. It is assumed that you have already installed the `kernel-headers` and `kernel-source` packages and that you issue all commands from the `/usr/src/linux/` directory.

It is important to begin a kernel build with the source tree in a known condition. Therefore, it is recommended that you begin with the command `make mrproper`. This will remove any configuration files along with the remains of any previous builds that may be scattered around the source tree. Now you must create a configuration file that will determine which components to include in your new kernel. Depending upon your hardware and personal preferences there are three methods available to configure the kernel.

- `make config` An interactive text program. Components are presented and you answer with `Y` (yes), `N` (no), or `M` (module).
- `make menuconfig` A graphic, menu driven program. Components are presented in a menu of categories, you select the desired components in the same manner used in the Red Hat Linux installation program. Toggle the tag corresponding to the item you want included; **Y** (yes), **N** (no), or **M** (module).
- `make xconfig` An X Windows program. Components are listed in different levels of menus, components are selected using a mouse. Again, select **Y** (yes), **N** (no), or **M** (module).

Linux Redhat, at 6.1.1



The next step consists of the actual compilation of the source code components into a working program that your machine can use to boot. The method described here is the easiest to recover from in the event of a mishap. If you are interested in other possibilities details can be found in the Kernel-HOWTO or in the `Makefile` in `/usr/src/linux` on your Linux system.

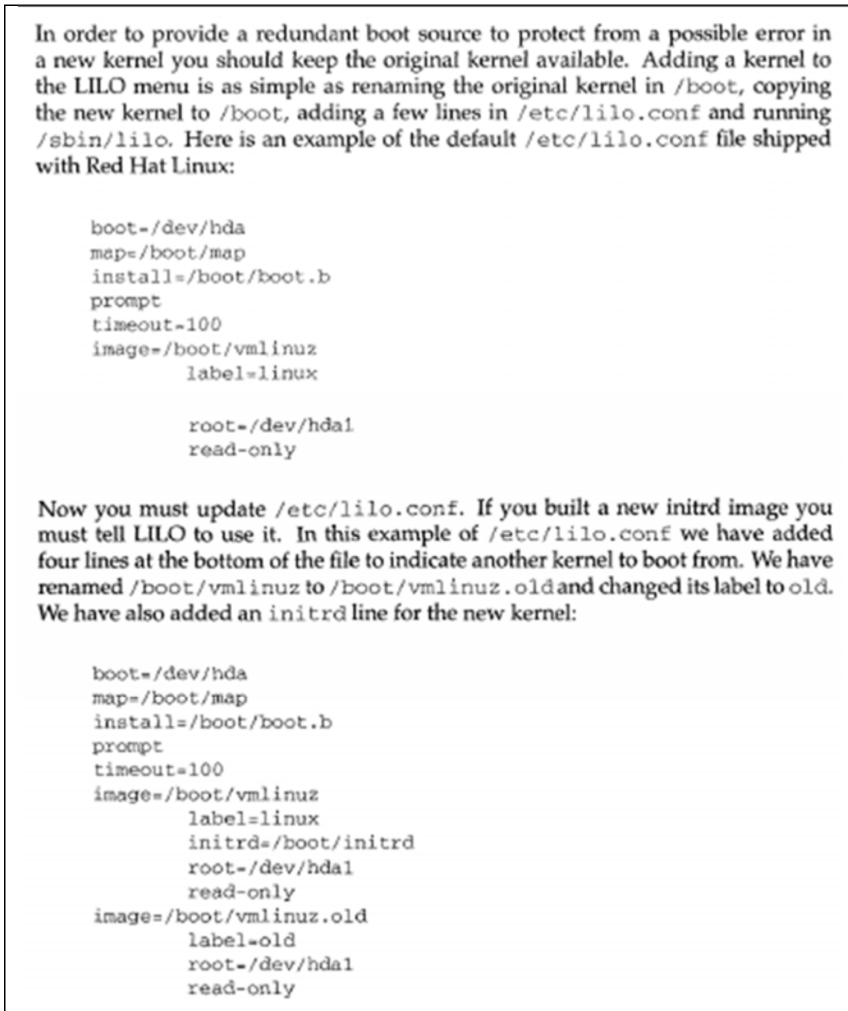
- Build the kernel with `make boot`.
- Build any modules you configured with `make modules`.
- Move the old set of modules out of the way with:

```
rm -rf /lib/modules/2.0.29-old
mv /lib/modules/2.0.29 /lib/modules/2.0.29-old
```

  Of course, if you have upgraded your kernel, replace `2.0.29` with the version you are using.

- Install the new modules (even if you didn't build any) with `make modules_install`.

Linux Redhat, at 6.1.1



In order to provide a redundant boot source to protect from a possible error in a new kernel you should keep the original kernel available. Adding a kernel to the LILO menu is as simple as renaming the original kernel in `/boot`, copying the new kernel to `/boot`, adding a few lines in `/etc/lilo.conf` and running `/sbin/lilo`. Here is an example of the default `/etc/lilo.conf` file shipped with Red Hat Linux:

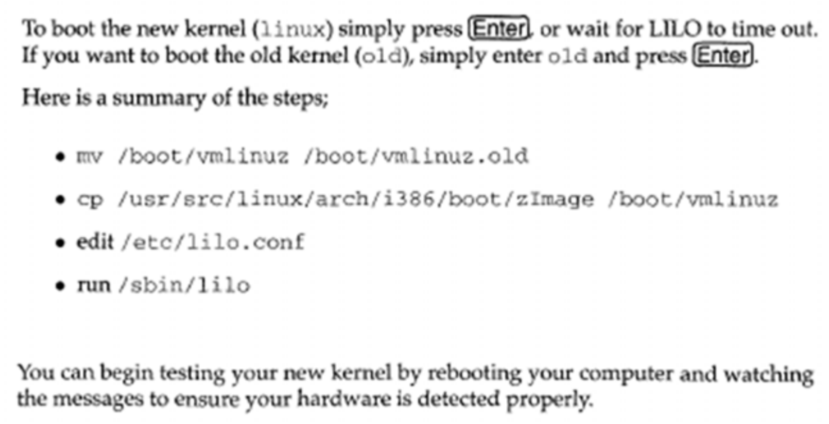
```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=100
image=/boot/vmlinuz
        label=linux

        root=/dev/hda1
        read-only
```

Now you must update `/etc/lilo.conf`. If you built a new initrd image you must tell LILO to use it. In this example of `/etc/lilo.conf` we have added four lines at the bottom of the file to indicate another kernel to boot from. We have renamed `/boot/vmlinuz` to `/boot/vmlinuz.old` and changed its label to `old`. We have also added an `initrd` line for the new kernel:

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=100
image=/boot/vmlinuz
        label=linux
        initrd=/boot/initrd
        root=/dev/hda1
        read-only
image=/boot/vmlinuz.old
        label=old
        root=/dev/hda1
        read-only
```

Linux Redhat, at 6.1.1



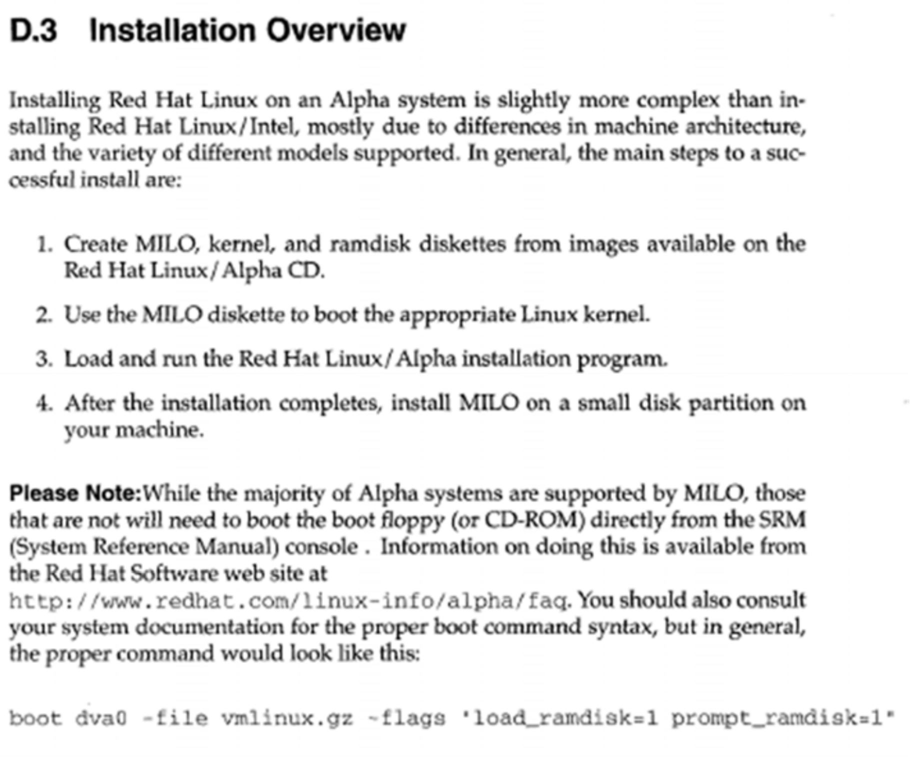
To boot the new kernel (`linux`) simply press Enter, or wait for LILO to time out. If you want to boot the old kernel (`old`), simply enter `old` and press Enter.

Here is a summary of the steps;

- `mv /boot/vmlinuz /boot/vmlinuz.old`
- `cp /usr/src/linux/arch/i386/boot/zImage /boot/vmlinuz`
- edit `/etc/lilo.conf`
- run `/sbin/lilo`

You can begin testing your new kernel by rebooting your computer and watching the messages to ensure your hardware is detected properly.

Linux Redhat, at 6.1.1



D.3 Installation Overview

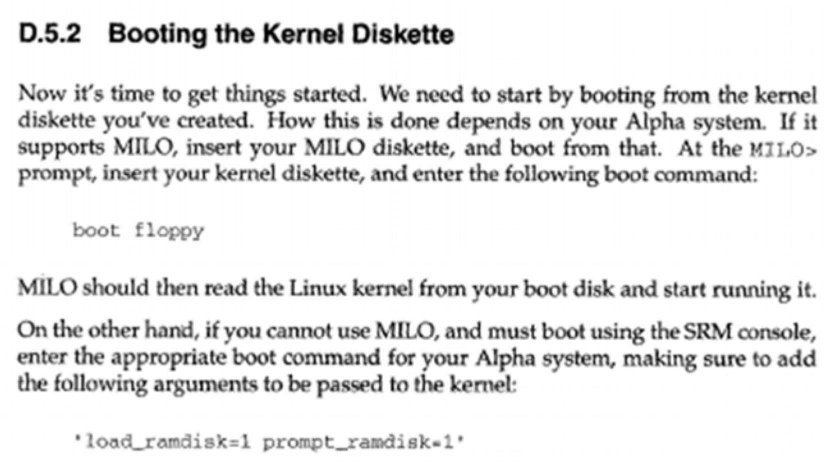
Installing Red Hat Linux on an Alpha system is slightly more complex than installing Red Hat Linux/Intel, mostly due to differences in machine architecture, and the variety of different models supported. In general, the main steps to a successful install are:

1. Create MILO, kernel, and ramdisk diskettes from images available on the Red Hat Linux/Alpha CD.
2. Use the MILO diskette to boot the appropriate Linux kernel.
3. Load and run the Red Hat Linux/Alpha installation program.
4. After the installation completes, install MILO on a small disk partition on your machine.

Please Note: While the majority of Alpha systems are supported by MILO, those that are not will need to boot the boot floppy (or CD-ROM) directly from the SRM (System Reference Manual) console . Information on doing this is available from the Red Hat Software web site at `http://www.redhat.com/linux-info/alpha/faq`. You should also consult your system documentation for the proper boot command syntax, but in general, the proper command would look like this:

```
boot dva0 -file vmlinux.gz -flags 'load_ramdisk=1 prompt_ramdisk=1'
```

Linux Redhat, at Appendix D.3



D.5.2 Booting the Kernel Diskette

Now it's time to get things started. We need to start by booting from the kernel diskette you've created. How this is done depends on your Alpha system. If it supports MILO, insert your MILO diskette, and boot from that. At the `MILO>` prompt, insert your kernel diskette, and enter the following boot command:

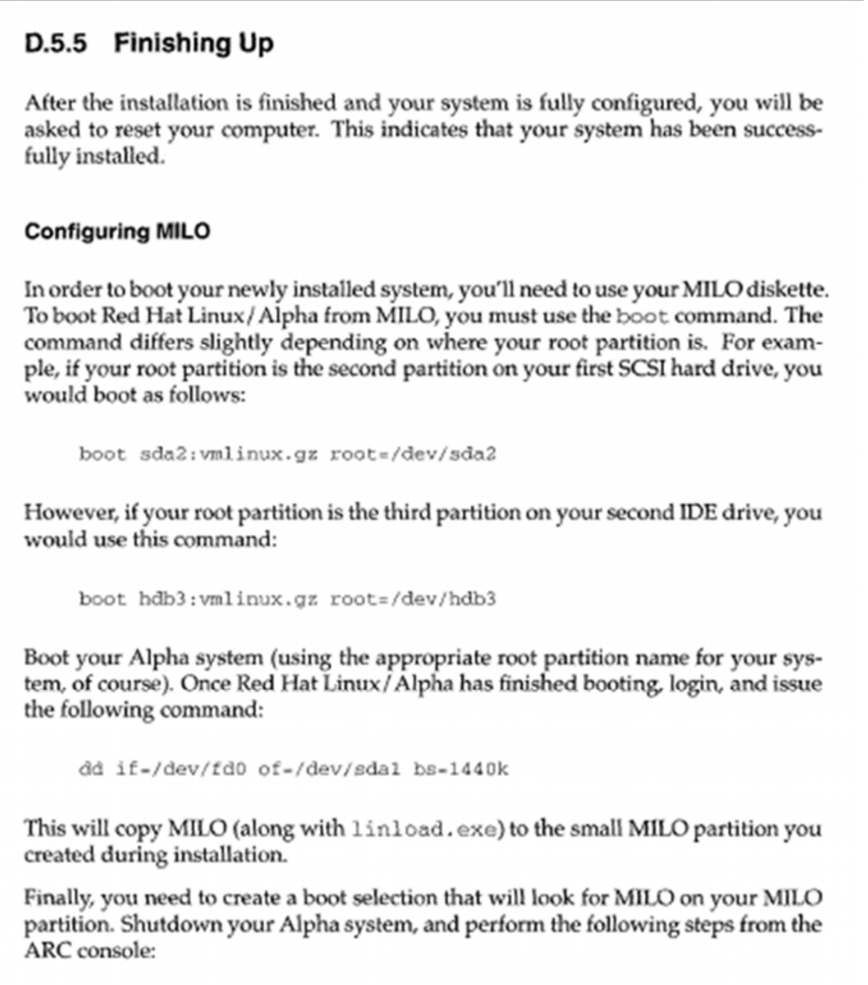
```
    boot floppy
```

MILO should then read the Linux kernel from your boot disk and start running it.

On the other hand, if you cannot use MILO, and must boot using the SRM console, enter the appropriate boot command for your Alpha system, making sure to add the following arguments to be passed to the kernel:

```
    'load_ramdisk=1 prompt_ramdisk=1'
```

Linux Redhat, at Appendix D.5.2



D.5.5 Finishing Up

After the installation is finished and your system is fully configured, you will be asked to reset your computer. This indicates that your system has been successfully installed.

Configuring MILO

In order to boot your newly installed system, you'll need to use your MILO diskette. To boot Red Hat Linux/Alpha from MILO, you must use the `boot` command. The command differs slightly depending on where your root partition is. For example, if your root partition is the second partition on your first SCSI hard drive, you would boot as follows:

```
boot sda2:vmlinux.gz root=/dev/sda2
```

However, if your root partition is the third partition on your second IDE drive, you would use this command:

```
boot hdb3:vmlinux.gz root=/dev/hdb3
```

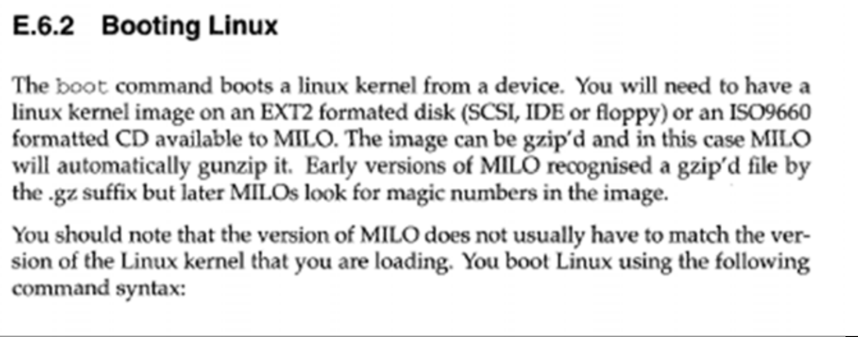
Boot your Alpha system (using the appropriate root partition name for your system, of course). Once Red Hat Linux/Alpha has finished booting, login, and issue the following command:

```
dd if-/dev/fd0 of-/dev/sda1 bs-1440k
```

This will copy MILO (along with `linload.exe`) to the small MILO partition you created during installation.

Finally, you need to create a boot selection that will look for MILO on your MILO partition. Shutdown your Alpha system, and perform the following steps from the ARC console:

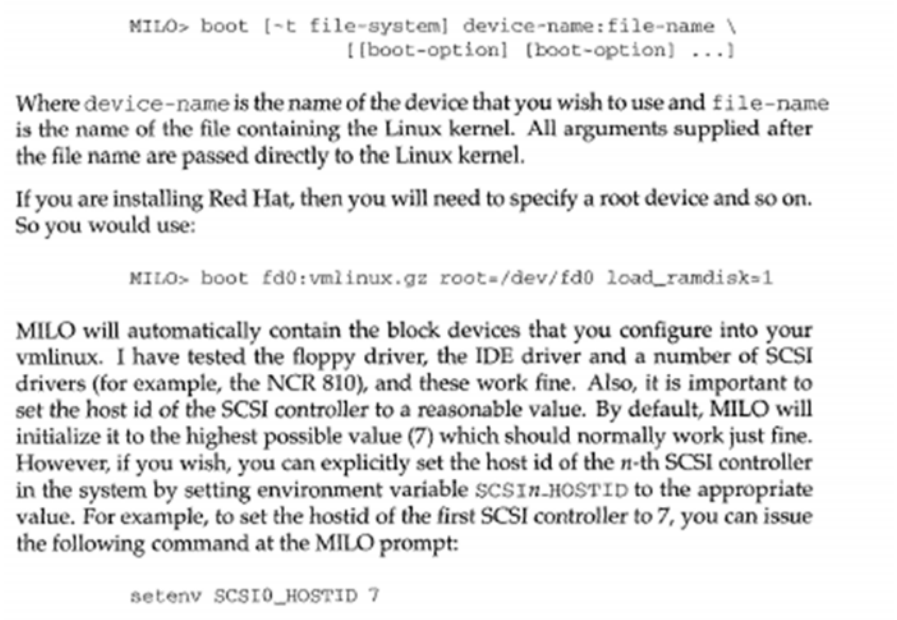
Linux Redhat, at Appendix D5.5



E.6.2 Booting Linux

The `boot` command boots a linux kernel from a device. You will need to have a linux kernel image on an EXT2 formated disk (SCSI, IDE or floppy) or an ISO9660 formatted CD available to MILO. The image can be gzip'd and in this case MILO will automatically gunzip it. Early versions of MILO recognised a gzip'd file by the .gz suffix but later MILOs look for magic numbers in the image.

You should note that the version of MILO does not usually have to match the version of the Linux kernel that you are loading. You boot Linux using the following command syntax:



```
MILO> boot [-t file-system] device-name:file-name \
                  [[boot-option] [boot-option] ...]
```

Where `device-name` is the name of the device that you wish to use and `file-name` is the name of the file containing the Linux kernel. All arguments supplied after the file name are passed directly to the Linux kernel.

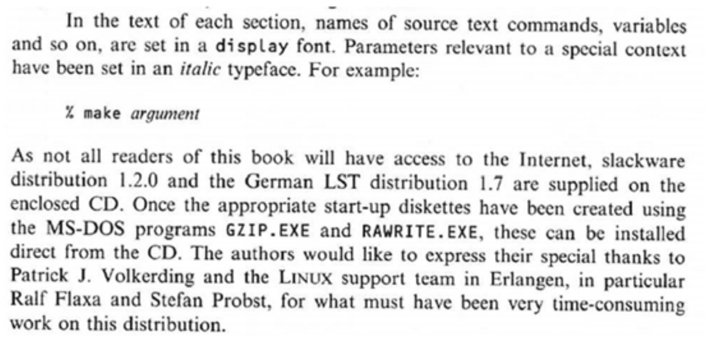
If you are installing Red Hat, then you will need to specify a root device and so on. So you would use:

```
MILO> boot fd0:vmlinux.gz root=/dev/fd0 load_ramdisk=1
```

MILO will automatically contain the block devices that you configure into your vmlinux. I have tested the floppy driver, the IDE driver and a number of SCSI drivers (for example, the NCR 810), and these work fine. Also, it is important to set the host id of the SCSI controller to a reasonable value. By default, MILO will initialize it to the highest possible value (7) which should normally work just fine. However, if you wish, you can explicitly set the host id of the *n*-th SCSI controller in the system by setting environment variable `SCSIn_HOSTID` to the appropriate value. For example, to set the hostid of the first SCSI controller to 7, you can issue the following command at the MILO prompt:

```
setenv SCSI0_HOSTID 7
```

Linux Redhat, at Appendix E.6.2

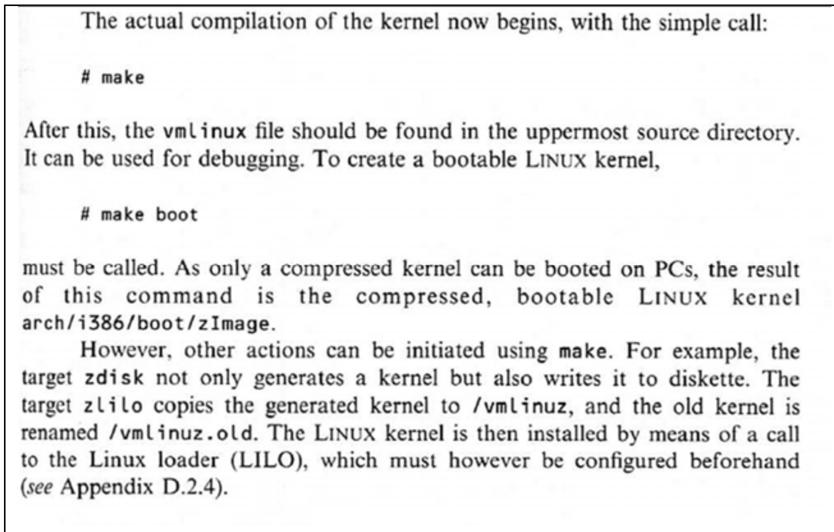


In the text of each section, names of source text commands, variables and so on, are set in a `display` font. Parameters relevant to a special context have been set in an *italic* typeface. For example:

```
% make argument
```

As not all readers of this book will have access to the Internet, slackware distribution 1.2.0 and the German LST distribution 1.7 are supplied on the enclosed CD. Once the appropriate start-up diskettes have been created using the MS-DOS programs **GZIP.EXE** and **RAWRITE.EXE**, these can be installed direct from the CD. The authors would like to express their special thanks to Patrick J. Volkerding and the LINUX support team in Erlangen, in particular Ralf Flaxa and Stefan Probst, for what must have been very time-consuming work on this distribution.

Beck, at Preface


The actual compilation of the kernel now begins, with the simple call:

```
# make
```

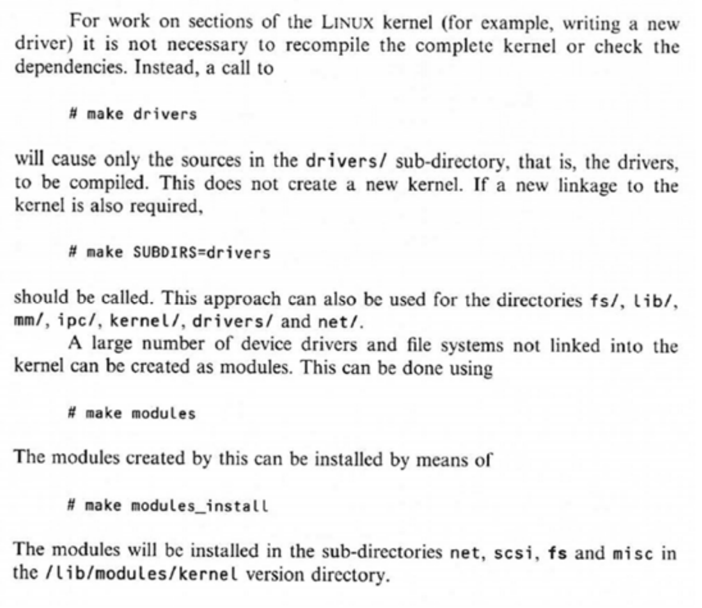
After this, the `vmlinux` file should be found in the uppermost source directory. It can be used for debugging. To create a bootable LINUX kernel,

```
# make boot
```

must be called. As only a compressed kernel can be booted on PCs, the result of this command is the compressed, bootable LINUX kernel `arch/i386/boot/zImage`.

However, other actions can be initiated using `make`. For example, the target `zdisk` not only generates a kernel but also writes it to diskette. The target `zlilo` copies the generated kernel to `/vmlinuz`, and the old kernel is renamed `/vmlinuz.old`. The LINUX kernel is then installed by means of a call to the Linux loader (LILO), which must however be configured beforehand (*see* Appendix D.2.4).

Beck, at 2.2


For work on sections of the LINUX kernel (for example, writing a new driver) it is not necessary to recompile the complete kernel or check the dependencies. Instead, a call to

```
# make drivers
```

will cause only the sources in the `drivers/` sub-directory, that is, the drivers, to be compiled. This does not create a new kernel. If a new linkage to the kernel is also required,

```
# make SUBDIRS=drivers
```

should be called. This approach can also be used for the directories `fs/`, `lib/`, `mm/`, `ipc/`, `kernel/`, `drivers/` and `net/`.

A large number of device drivers and file systems not linked into the kernel can be created as modules. This can be done using

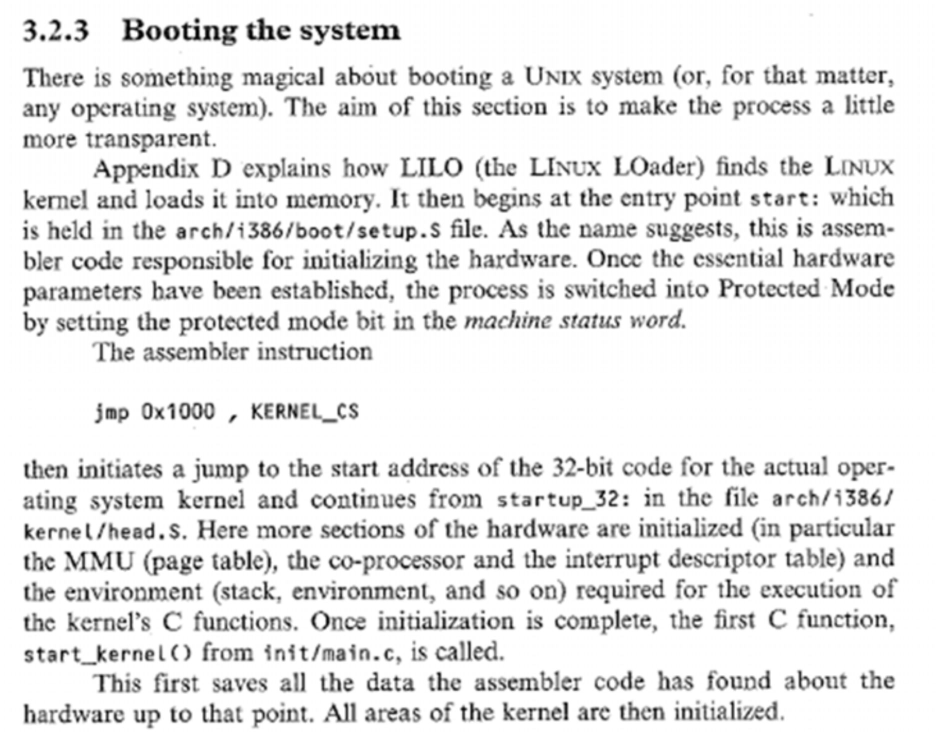
```
# make modules
```

The modules created by this can be installed by means of

```
# make modules_install
```

The modules will be installed in the sub-directories `net`, `scsi`, `fs` and `misc` in the `/lib/modules/kernel` version directory.

Beck, at 2.2



3.2.3 Booting the system

There is something magical about booting a UNIX system (or, for that matter, any operating system). The aim of this section is to make the process a little more transparent.

Appendix D explains how LILO (the LINUX LOader) finds the LINUX kernel and loads it into memory. It then begins at the entry point `start:` which is held in the `arch/i386/boot/setup.S` file. As the name suggests, this is assembler code responsible for initializing the hardware. Once the essential hardware parameters have been established, the process is switched into Protected Mode by setting the protected mode bit in the *machine status word.*

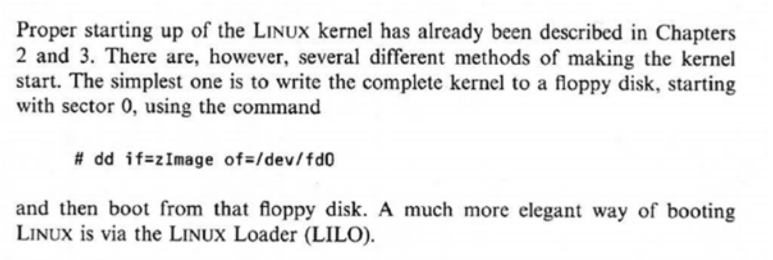
The assembler instruction

```
jmp 0x1000 , KERNEL_CS
```

then initiates a jump to the start address of the 32-bit code for the actual operating system kernel and continues from `startup_32:` in the file `arch/i386/kernel/head.S`. Here more sections of the hardware are initialized (in particular the MMU (page table), the co-processor and the interrupt descriptor table) and the environment (stack, environment, and so on) required for the execution of the kernel's C functions. Once initialization is complete, the first C function, `start_kernel()` from `init/main.c`, is called.

This first saves all the data the assembler code has found about the hardware up to that point. All areas of the kernel are then initialized.

Beck, at 3.2.3

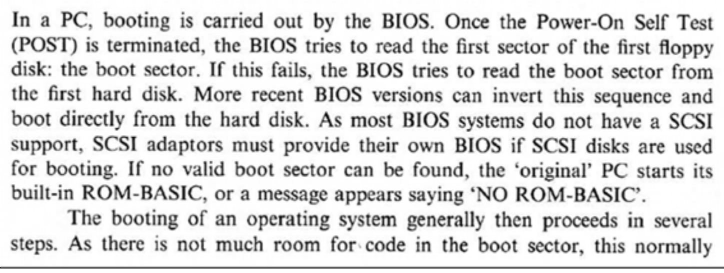


Proper starting up of the LINUX kernel has already been described in Chapters 2 and 3. There are, however, several different methods of making the kernel start. The simplest one is to write the complete kernel to a floppy disk, starting with sector 0, using the command

```
# dd if=zImage of=/dev/fd0
```

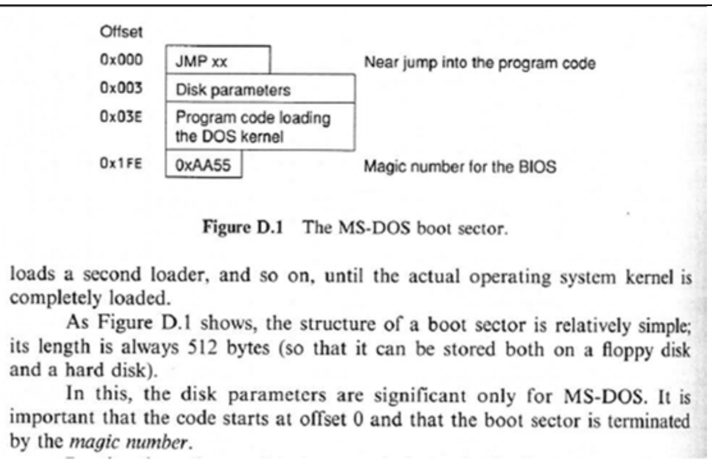
and then boot from that floppy disk. A much more elegant way of booting LINUX is via the LINUX Loader (LILO).

Beck, at Appendix D



In a PC, booting is carried out by the BIOS. Once the Power-On Self Test (POST) is terminated, the BIOS tries to read the first sector of the first floppy disk: the boot sector. If this fails, the BIOS tries to read the boot sector from the first hard disk. More recent BIOS versions can invert this sequence and boot directly from the hard disk. As most BIOS systems do not have a SCSI support, SCSI adaptors must provide their own BIOS if SCSI disks are used for booting. If no valid boot sector can be found, the 'original' PC starts its built-in ROM-BASIC, or a message appears saying 'NO ROM-BASIC'.

The booting of an operating system generally then proceeds in several steps. As there is not much room for code in the boot sector, this normally



Offset		
0x000	JMP xx	Near jump into the program code
0x003	Disk parameters	
0x03E	Program code loading the DOS kernel	
0x1FE	0xAA55	Magic number for the BIOS

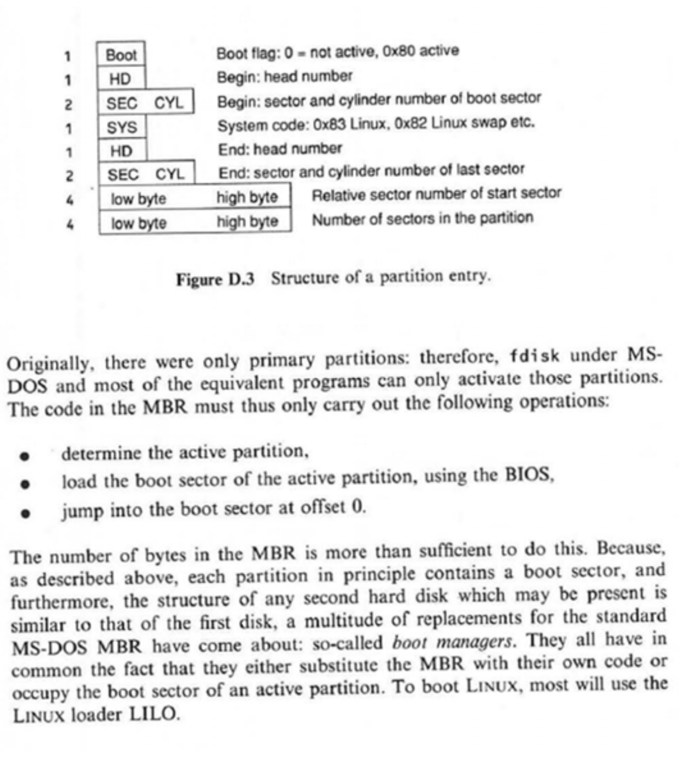
Figure D.1 The MS-DOS boot sector.

loads a second loader, and so on, until the actual operating system kernel is completely loaded.

As Figure D.1 shows, the structure of a boot sector is relatively simple; its length is always 512 bytes (so that it can be stored both on a floppy disk and a hard disk).

In this, the disk parameters are significant only for MS-DOS. It is important that the code starts at offset 0 and that the boot sector is terminated by the *magic number*.

Beck, at Appendix D.1



1	Boot	Boot flag: 0 = not active, 0x80 active
1	HD	Begin: head number
2	SEC CYL	Begin: sector and cylinder number of boot sector
1	SYS	System code: 0x83 Linux, 0x82 Linux swap etc.
1	HD	End: head number
2	SEC CYL	End: sector and cylinder number of last sector
4	low byte high byte	Relative sector number of start sector
4	low byte high byte	Number of sectors in the partition

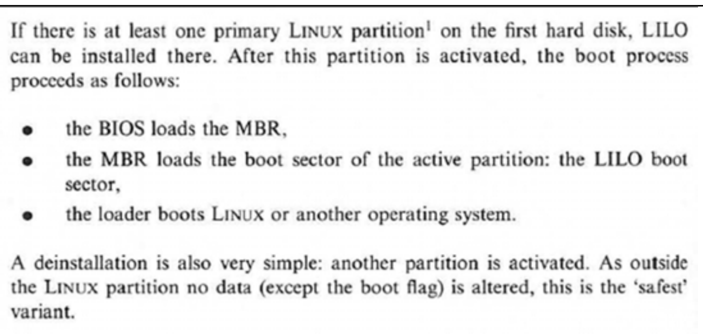
Figure D.3 Structure of a partition entry.

Originally, there were only primary partitions: therefore, `fdisk` under MS-DOS and most of the equivalent programs can only activate those partitions. The code in the MBR must thus only carry out the following operations:

- determine the active partition,
- load the boot sector of the active partition, using the BIOS,
- jump into the boot sector at offset 0.

The number of bytes in the MBR is more than sufficient to do this. Because, as described above, each partition in principle contains a boot sector, and furthermore, the structure of any second hard disk which may be present is similar to that of the first disk, a multitude of replacements for the standard MS-DOS MBR have come about: so-called *boot managers*. They all have in common the fact that they either substitute the MBR with their own code or occupy the boot sector of an active partition. To boot LINUX, most will use the LINUX loader LILO.

Beck, at Appendix D.1

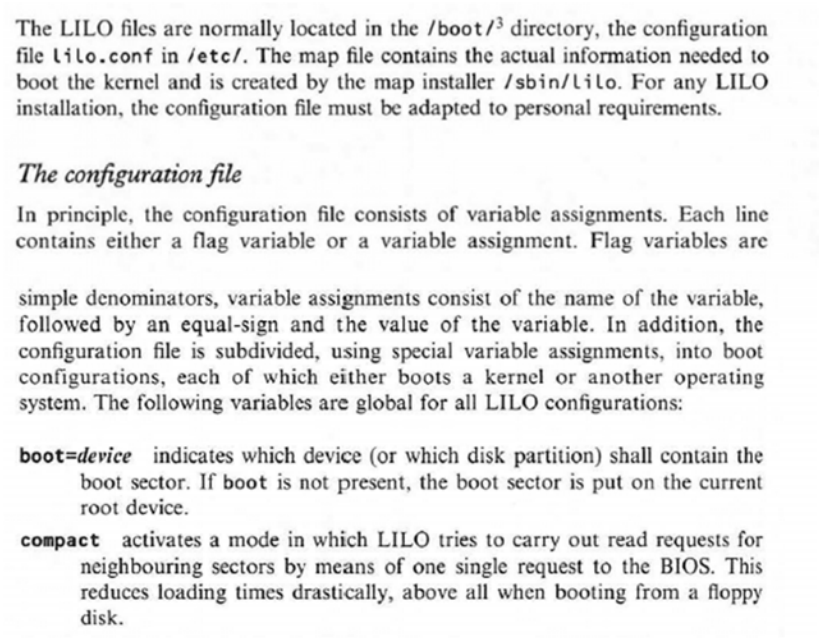


If there is at least one primary LINUX partition[1] on the first hard disk, LILO can be installed there. After this partition is activated, the boot process proceeds as follows:

- the BIOS loads the MBR,
- the MBR loads the boot sector of the active partition: the LILO boot sector,
- the loader boots LINUX or another operating system.

A deinstallation is also very simple: another partition is activated. As outside the LINUX partition no data (except the boot flag) is altered, this is the 'safest' variant.

Beck, at Appendix D.2.1



The LILO files are normally located in the `/boot/`[3] directory, the configuration file `lilo.conf` in `/etc/`. The map file contains the actual information needed to boot the kernel and is created by the map installer `/sbin/lilo`. For any LILO installation, the configuration file must be adapted to personal requirements.

The configuration file

In principle, the configuration file consists of variable assignments. Each line contains either a flag variable or a variable assignment. Flag variables are

simple denominators, variable assignments consist of the name of the variable, followed by an equal-sign and the value of the variable. In addition, the configuration file is subdivided, using special variable assignments, into boot configurations, each of which either boots a kernel or another operating system. The following variables are global for all LILO configurations:

`boot=`*`device`* indicates which device (or which disk partition) shall contain the boot sector. If **`boot`** is not present, the boot sector is put on the current root device.

`compact` activates a mode in which LILO tries to carry out read requests for neighbouring sectors by means of one single request to the BIOS. This reduces loading times drastically, above all when booting from a floppy disk.

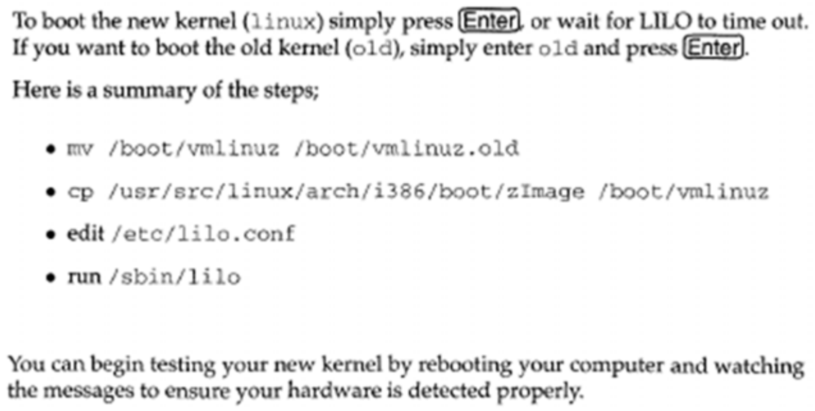
Beck, at Appendix D.2.4

1.2 initializing a central processing unit of said computer system;	Linux Kernel, as evidenced by the example citations below, discloses "initializing a central processing unit of said computer system;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, initializing a central processing unit of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Linux Kernel discloses this limitation:

Linux Kernel discloses this limitation:

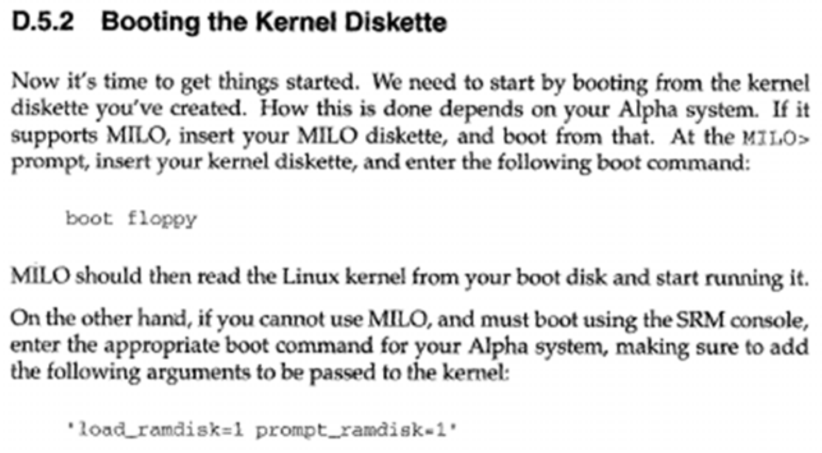

To boot the new kernel (`linux`) simply press Enter, or wait for LILO to time out. If you want to boot the old kernel (`old`), simply enter `old` and press Enter.

Here is a summary of the steps;

- `mv /boot/vmlinuz /boot/vmlinuz.old`
- `cp /usr/src/linux/arch/i386/boot/zImage /boot/vmlinuz`
- edit `/etc/lilo.conf`
- run `/sbin/lilo`

You can begin testing your new kernel by rebooting your computer and watching the messages to ensure your hardware is detected properly.

Linux Redhat, at 6.1.1


D.5.2 Booting the Kernel Diskette

Now it's time to get things started. We need to start by booting from the kernel diskette you've created. How this is done depends on your Alpha system. If it supports MILO, insert your MILO diskette, and boot from that. At the `MILO>` prompt, insert your kernel diskette, and enter the following boot command:

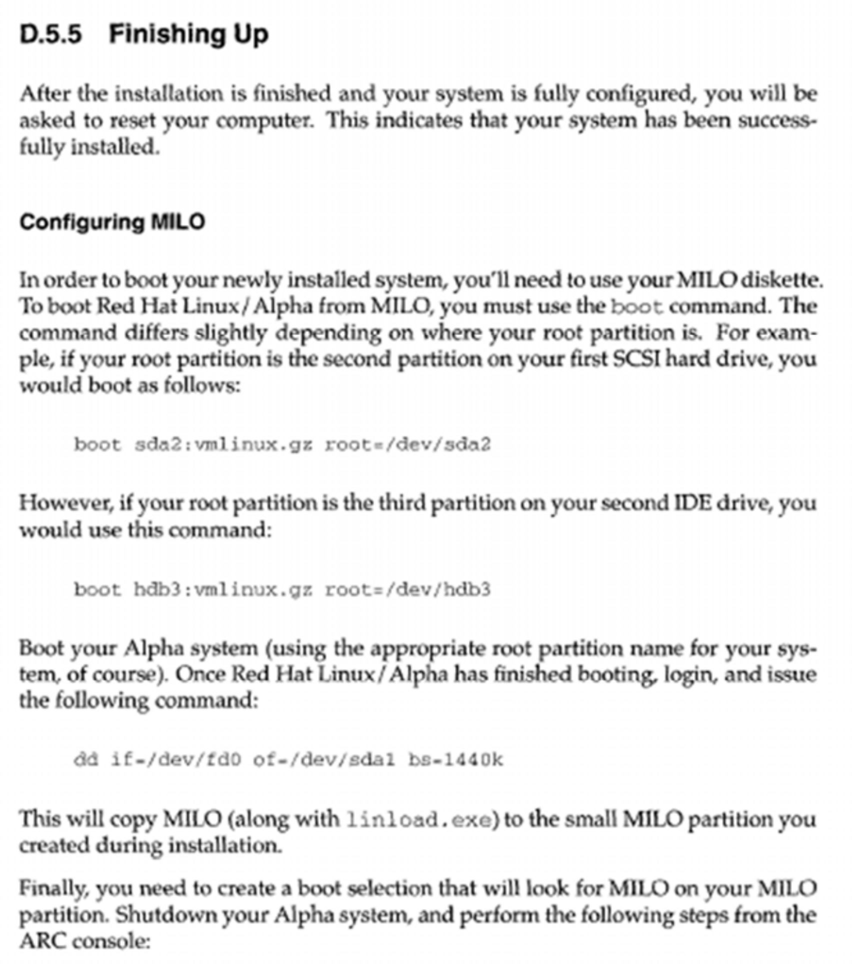
```
boot floppy
```

MILO should then read the Linux kernel from your boot disk and start running it.

On the other hand, if you cannot use MILO, and must boot using the SRM console, enter the appropriate boot command for your Alpha system, making sure to add the following arguments to be passed to the kernel:

```
'load_ramdisk=1 prompt_ramdisk=1'
```

Linux Redhat, at Appendix D.5.2



D.5.5 Finishing Up

After the installation is finished and your system is fully configured, you will be asked to reset your computer. This indicates that your system has been successfully installed.

Configuring MILO

In order to boot your newly installed system, you'll need to use your MILO diskette. To boot Red Hat Linux/Alpha from MILO, you must use the `boot` command. The command differs slightly depending on where your root partition is. For example, if your root partition is the second partition on your first SCSI hard drive, you would boot as follows:

```
boot sda2:vmlinux.gz root=/dev/sda2
```

However, if your root partition is the third partition on your second IDE drive, you would use this command:

```
boot hdb3:vmlinux.gz root=/dev/hdb3
```

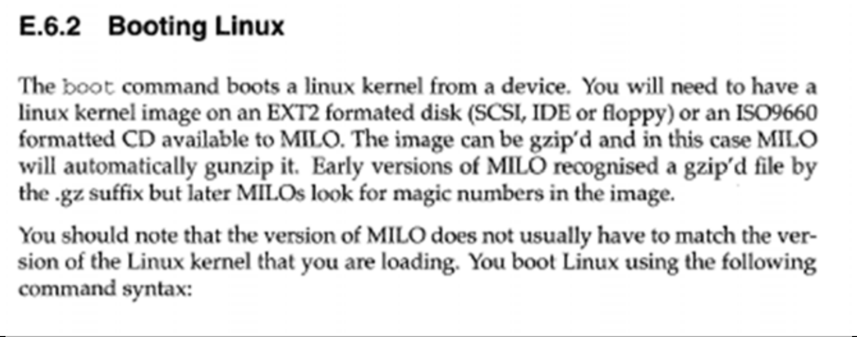
Boot your Alpha system (using the appropriate root partition name for your system, of course). Once Red Hat Linux/Alpha has finished booting, login, and issue the following command:

```
dd if=/dev/fd0 of=/dev/sda1 bs=1440k
```

This will copy MILO (along with `linload.exe`) to the small MILO partition you created during installation.

Finally, you need to create a boot selection that will look for MILO on your MILO partition. Shutdown your Alpha system, and perform the following steps from the ARC console:

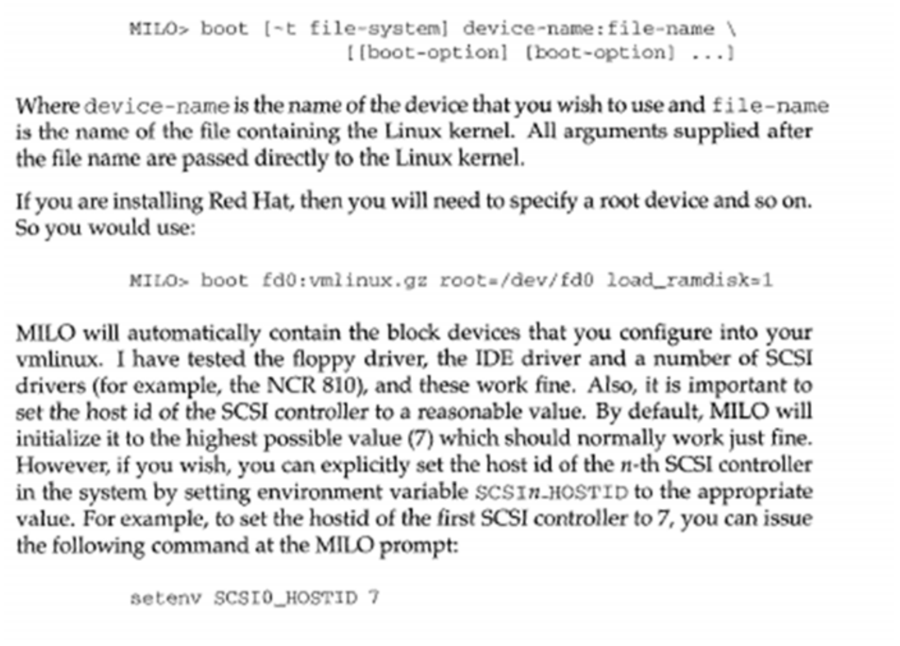
Linux Redhat, at Appendix D5.5



E.6.2 Booting Linux

The `boot` command boots a linux kernel from a device. You will need to have a linux kernel image on an EXT2 formated disk (SCSI, IDE or floppy) or an ISO9660 formatted CD available to MILO. The image can be gzip'd and in this case MILO will automatically gunzip it. Early versions of MILO recognised a gzip'd file by the .gz suffix but later MILOs look for magic numbers in the image.

You should note that the version of MILO does not usually have to match the version of the Linux kernel that you are loading. You boot Linux using the following command syntax:



```
MILO> boot [-t file-system] device-name:file-name \
                    [[boot-option] [boot-option] ...]
```

Where `device-name` is the name of the device that you wish to use and `file-name` is the name of the file containing the Linux kernel. All arguments supplied after the file name are passed directly to the Linux kernel.

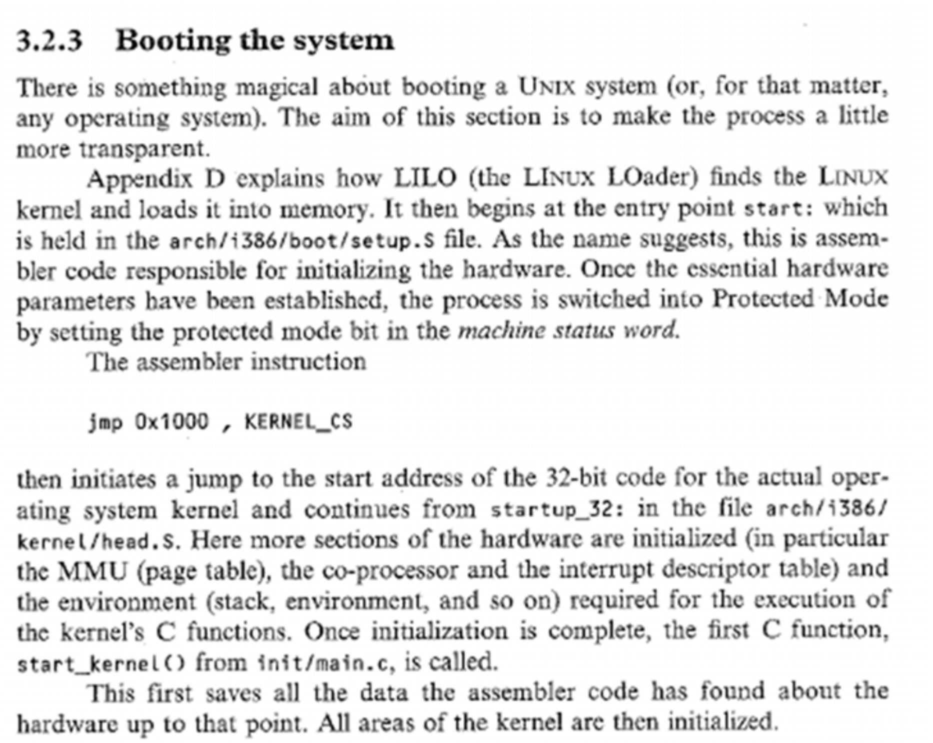
If you are installing Red Hat, then you will need to specify a root device and so on. So you would use:

```
MILO> boot fd0:vmlinux.gz root=/dev/fd0 load_ramdisk=1
```

MILO will automatically contain the block devices that you configure into your vmlinux. I have tested the floppy driver, the IDE driver and a number of SCSI drivers (for example, the NCR 810), and these work fine. Also, it is important to set the host id of the SCSI controller to a reasonable value. By default, MILO will initialize it to the highest possible value (7) which should normally work just fine. However, if you wish, you can explicitly set the host id of the *n*-th SCSI controller in the system by setting environment variable `SCSIn_HOSTID` to the appropriate value. For example, to set the hostid of the first SCSI controller to 7, you can issue the following command at the MILO prompt:

```
setenv SCSI0_HOSTID 7
```

Linux Redhat, at Appendix E.6.2



3.2.3 Booting the system

There is something magical about booting a UNIX system (or, for that matter, any operating system). The aim of this section is to make the process a little more transparent.

Appendix D explains how LILO (the LINUX LOader) finds the LINUX kernel and loads it into memory. It then begins at the entry point `start:` which is held in the `arch/i386/boot/setup.S` file. As the name suggests, this is assembler code responsible for initializing the hardware. Once the essential hardware parameters have been established, the process is switched into Protected Mode by setting the protected mode bit in the *machine status word*.

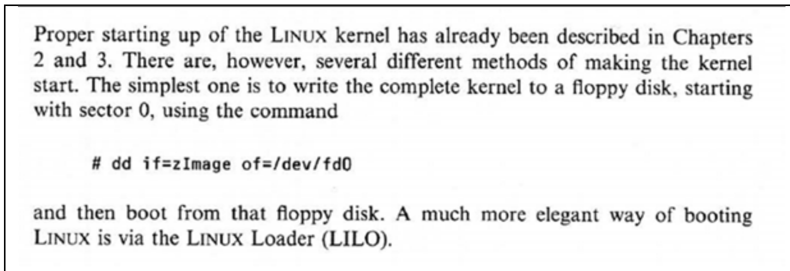
The assembler instruction

```
jmp 0x1000 , KERNEL_CS
```

then initiates a jump to the start address of the 32-bit code for the actual operating system kernel and continues from `startup_32:` in the file `arch/i386/kernel/head.S`. Here more sections of the hardware are initialized (in particular the MMU (page table), the co-processor and the interrupt descriptor table) and the environment (stack, environment, and so on) required for the execution of the kernel's C functions. Once initialization is complete, the first C function, `start_kernel()` from `init/main.c`, is called.

This first saves all the data the assembler code has found about the hardware up to that point. All areas of the kernel are then initialized.

Beck, at 3.2.3

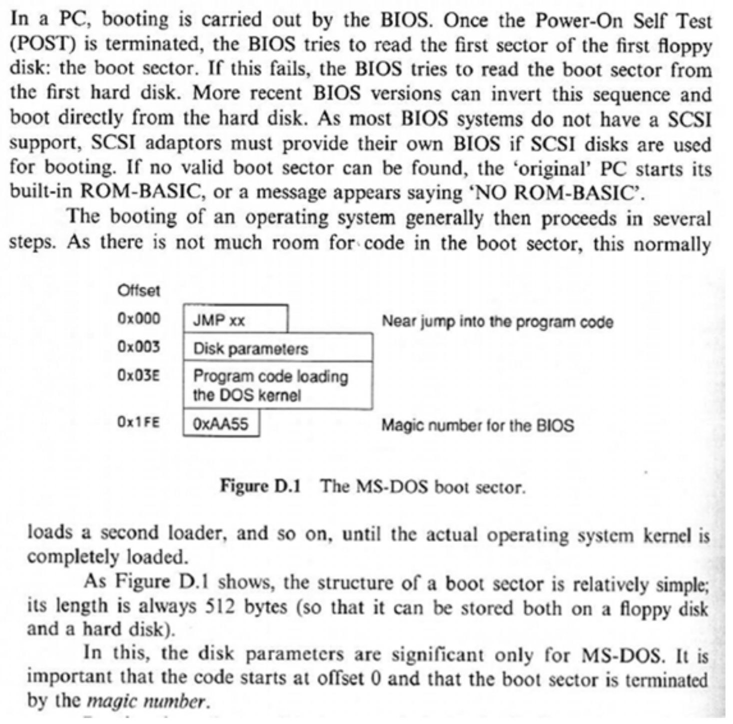


Proper starting up of the LINUX kernel has already been described in Chapters 2 and 3. There are, however, several different methods of making the kernel start. The simplest one is to write the complete kernel to a floppy disk, starting with sector 0, using the command

```
# dd if=zImage of=/dev/fd0
```

and then boot from that floppy disk. A much more elegant way of booting LINUX is via the LINUX Loader (LILO).

Beck, at Appendix D



In a PC, booting is carried out by the BIOS. Once the Power-On Self Test (POST) is terminated, the BIOS tries to read the first sector of the first floppy disk: the boot sector. If this fails, the BIOS tries to read the boot sector from the first hard disk. More recent BIOS versions can invert this sequence and boot directly from the hard disk. As most BIOS systems do not have a SCSI support, SCSI adaptors must provide their own BIOS if SCSI disks are used for booting. If no valid boot sector can be found, the 'original' PC starts its built-in ROM-BASIC, or a message appears saying 'NO ROM-BASIC'.

The booting of an operating system generally then proceeds in several steps. As there is not much room for code in the boot sector, this normally

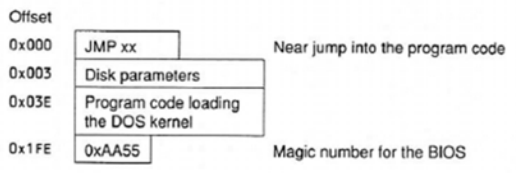

Offset
0x000
JMP xx
Near jump into the program code
0x003
Disk parameters
0x03E
Program code loading the DOS kernel
0x1FE
0xAA55
Magic number for the BIOS

Figure D.1 The MS-DOS boot sector.

loads a second loader, and so on, until the actual operating system kernel is completely loaded.

As Figure D.1 shows, the structure of a boot sector is relatively simple; its length is always 512 bytes (so that it can be stored both on a floppy disk and a hard disk).

In this, the disk parameters are significant only for MS-DOS. It is important that the code starts at offset 0 and that the boot sector is terminated by the *magic number*.

Beck, at Appendix D.1

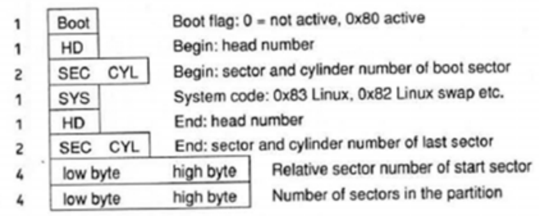

1 Boot — Boot flag: 0 = not active, 0x80 active
1 HD — Begin: head number
2 SEC CYL — Begin: sector and cylinder number of boot sector
1 SYS — System code: 0x83 Linux, 0x82 Linux swap etc.
1 HD — End: head number
2 SEC CYL — End: sector and cylinder number of last sector
4 low byte high byte — Relative sector number of start sector
4 low byte high byte — Number of sectors in the partition

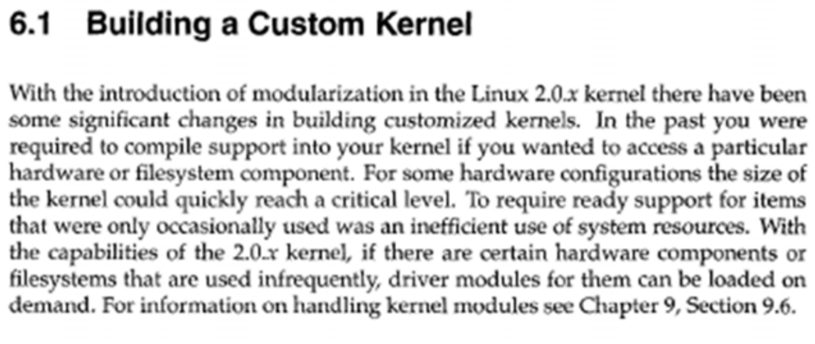
Figure D.3 Structure of a partition entry.

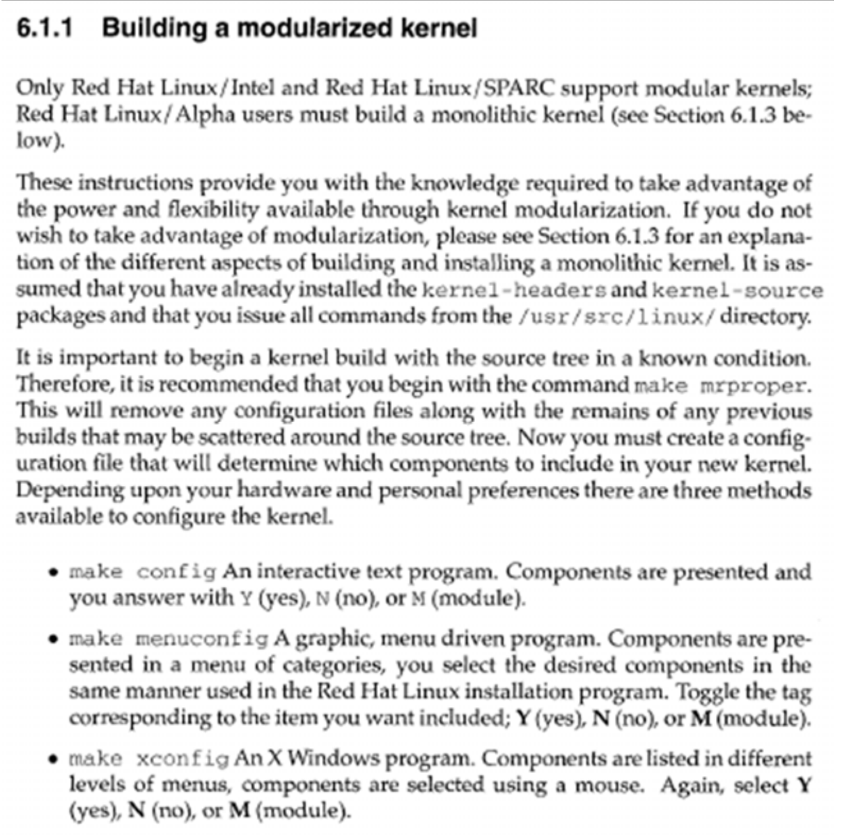
Originally, there were only primary partitions: therefore, `fdisk` under MS-DOS and most of the equivalent programs can only activate those partitions. The code in the MBR must thus only carry out the following operations:

- determine the active partition,
- load the boot sector of the active partition, using the BIOS,
- jump into the boot sector at offset 0.

The number of bytes in the MBR is more than sufficient to do this. Because, as described above, each partition in principle contains a boot sector, and furthermore, the structure of any second hard disk which may be present is similar to that of the first disk, a multitude of replacements for the standard MS-DOS MBR have come about: so-called *boot managers*. They all have in common the fact that they either substitute the MBR with their own code or occupy the boot sector of an active partition. To boot LINUX, most will use the LINUX loader LILO.

Beck, at Appendix D.1

1.3 preloading said at least a portion of said boot data in compressed form from said boot device to a memory;	Linux Kernel, as evidenced by the example citations below, discloses "preloading said at least a portion of said boot data in compressed form from said boot device to a memory;"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, preloading said at least a portion of said boot data in compressed form from said boot device to a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Linux Kernel discloses this limitation:  **6.1 Building a Custom Kernel** With the introduction of modularization in the Linux 2.0.*x* kernel there have been some significant changes in building customized kernels. In the past you were required to compile support into your kernel if you wanted to access a particular hardware or filesystem component. For some hardware configurations the size of the kernel could quickly reach a critical level. To require ready support for items that were only occasionally used was an inefficient use of system resources. With the capabilities of the 2.0.*x* kernel, if there are certain hardware components or filesystems that are used infrequently, driver modules for them can be loaded on demand. For information on handling kernel modules see Chapter 9, Section 9.6. Linux Redhat, at 6.1	



6.1.1 Building a modularized kernel

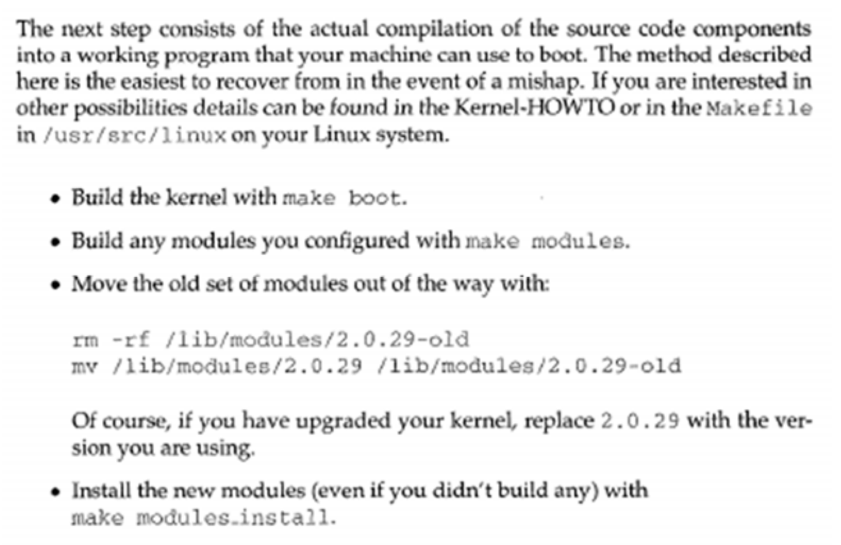
Only Red Hat Linux/Intel and Red Hat Linux/SPARC support modular kernels; Red Hat Linux/Alpha users must build a monolithic kernel (see Section 6.1.3 below).

These instructions provide you with the knowledge required to take advantage of the power and flexibility available through kernel modularization. If you do not wish to take advantage of modularization, please see Section 6.1.3 for an explanation of the different aspects of building and installing a monolithic kernel. It is assumed that you have already installed the `kernel-headers` and `kernel-source` packages and that you issue all commands from the `/usr/src/linux/` directory.

It is important to begin a kernel build with the source tree in a known condition. Therefore, it is recommended that you begin with the command `make mrproper`. This will remove any configuration files along with the remains of any previous builds that may be scattered around the source tree. Now you must create a configuration file that will determine which components to include in your new kernel. Depending upon your hardware and personal preferences there are three methods available to configure the kernel.

- `make config` An interactive text program. Components are presented and you answer with `Y` (yes), `N` (no), or `M` (module).
- `make menuconfig` A graphic, menu driven program. Components are presented in a menu of categories, you select the desired components in the same manner used in the Red Hat Linux installation program. Toggle the tag corresponding to the item you want included; **Y** (yes), **N** (no), or **M** (module).
- `make xconfig` An X Windows program. Components are listed in different levels of menus, components are selected using a mouse. Again, select **Y** (yes), **N** (no), or **M** (module).

Linux Redhat, at 6.1.1



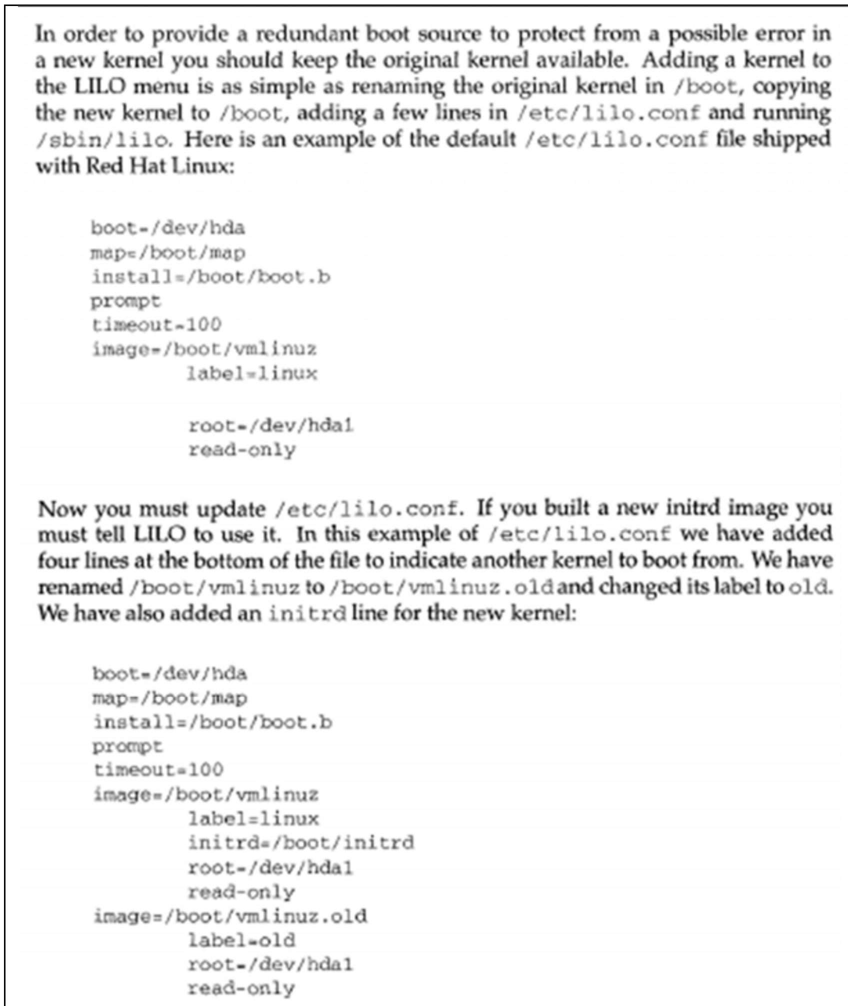
The next step consists of the actual compilation of the source code components into a working program that your machine can use to boot. The method described here is the easiest to recover from in the event of a mishap. If you are interested in other possibilities details can be found in the Kernel-HOWTO or in the `Makefile` in `/usr/src/linux` on your Linux system.

- Build the kernel with `make boot`.
- Build any modules you configured with `make modules`.
- Move the old set of modules out of the way with:

  ```
  rm -rf /lib/modules/2.0.29-old
  mv /lib/modules/2.0.29 /lib/modules/2.0.29-old
  ```

  Of course, if you have upgraded your kernel, replace `2.0.29` with the version you are using.
- Install the new modules (even if you didn't build any) with `make modules_install`.

Linux Redhat, at 6.1.1



In order to provide a redundant boot source to protect from a possible error in a new kernel you should keep the original kernel available. Adding a kernel to the LILO menu is as simple as renaming the original kernel in `/boot`, copying the new kernel to `/boot`, adding a few lines in `/etc/lilo.conf` and running `/sbin/lilo`. Here is an example of the default `/etc/lilo.conf` file shipped with Red Hat Linux:

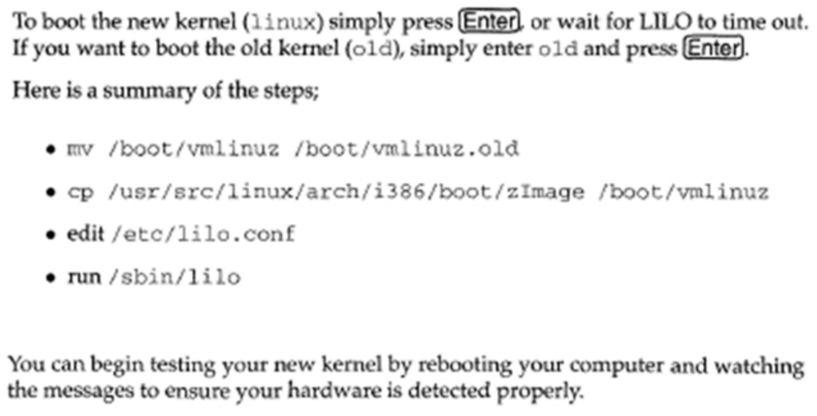
```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=100
image=/boot/vmlinuz
        label=linux

        root=/dev/hda1
        read-only
```

Now you must update `/etc/lilo.conf`. If you built a new initrd image you must tell LILO to use it. In this example of `/etc/lilo.conf` we have added four lines at the bottom of the file to indicate another kernel to boot from. We have renamed `/boot/vmlinuz` to `/boot/vmlinuz.old` and changed its label to `old`. We have also added an `initrd` line for the new kernel:

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=100
image=/boot/vmlinuz
        label=linux
        initrd=/boot/initrd
        root=/dev/hda1
        read-only
image=/boot/vmlinuz.old
        label=old
        root=/dev/hda1
        read-only
```

Linux Redhat, at 6.1.1



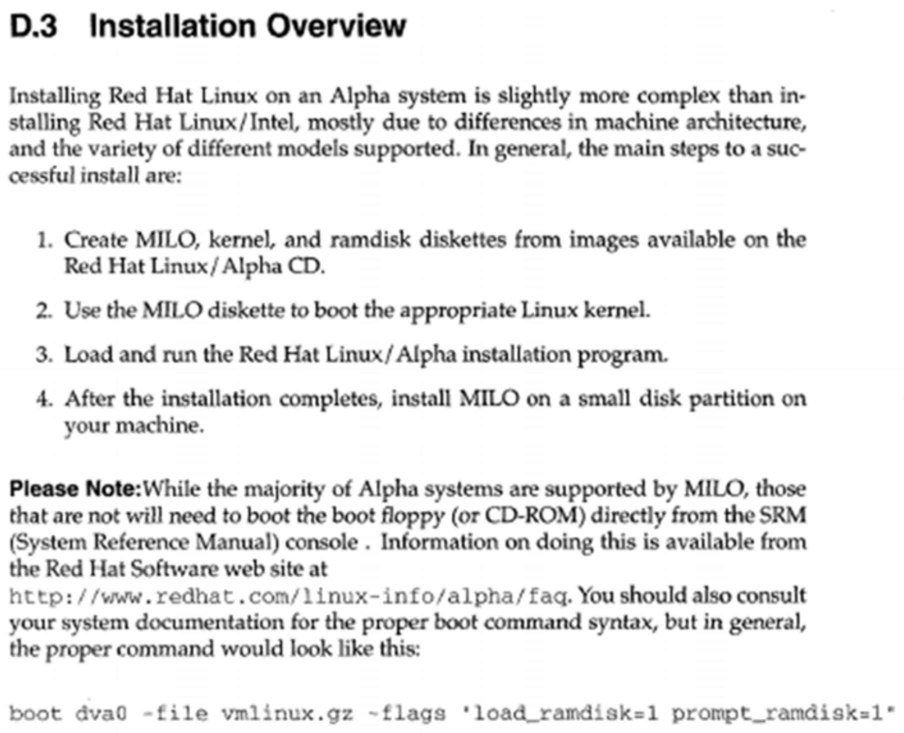
To boot the new kernel (`linux`) simply press Enter, or wait for LILO to time out. If you want to boot the old kernel (`old`), simply enter `old` and press Enter.

Here is a summary of the steps;

- `mv /boot/vmlinuz /boot/vmlinuz.old`
- `cp /usr/src/linux/arch/i386/boot/zImage /boot/vmlinuz`
- edit `/etc/lilo.conf`
- run `/sbin/lilo`

You can begin testing your new kernel by rebooting your computer and watching the messages to ensure your hardware is detected properly.

Linux Redhat, at 6.1.1



D.3 Installation Overview

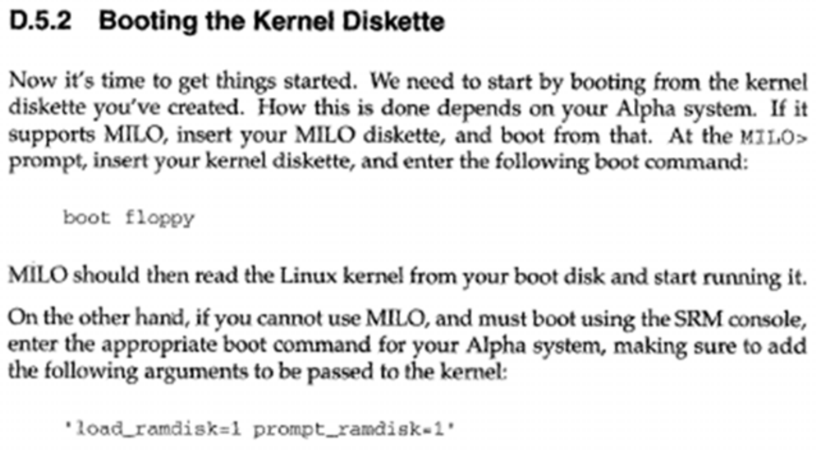
Installing Red Hat Linux on an Alpha system is slightly more complex than installing Red Hat Linux/Intel, mostly due to differences in machine architecture, and the variety of different models supported. In general, the main steps to a successful install are:

1. Create MILO, kernel, and ramdisk diskettes from images available on the Red Hat Linux/Alpha CD.
2. Use the MILO diskette to boot the appropriate Linux kernel.
3. Load and run the Red Hat Linux/Alpha installation program.
4. After the installation completes, install MILO on a small disk partition on your machine.

Please Note: While the majority of Alpha systems are supported by MILO, those that are not will need to boot the boot floppy (or CD-ROM) directly from the SRM (System Reference Manual) console . Information on doing this is available from the Red Hat Software web site at `http://www.redhat.com/linux-info/alpha/faq`. You should also consult your system documentation for the proper boot command syntax, but in general, the proper command would look like this:

```
boot dva0 -file vmlinux.gz -flags 'load_ramdisk=1 prompt_ramdisk=1'
```

Linux Redhat, at Appendix D.3



D.5.2 Booting the Kernel Diskette

Now it's time to get things started. We need to start by booting from the kernel diskette you've created. How this is done depends on your Alpha system. If it supports MILO, insert your MILO diskette, and boot from that. At the `MILO>` prompt, insert your kernel diskette, and enter the following boot command:

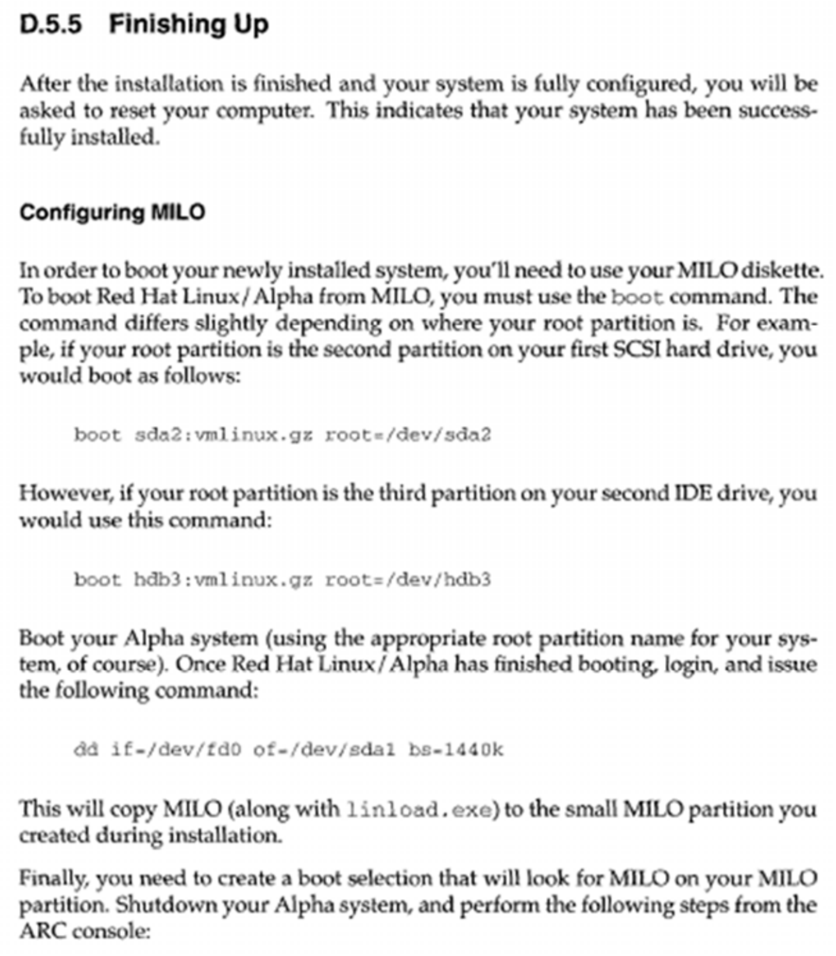
```
boot floppy
```

MILO should then read the Linux kernel from your boot disk and start running it.

On the other hand, if you cannot use MILO, and must boot using the SRM console, enter the appropriate boot command for your Alpha system, making sure to add the following arguments to be passed to the kernel:

```
'load_ramdisk=1 prompt_ramdisk=1'
```

Linux Redhat, at Appendix D.5.2



D.5.5 Finishing Up

After the installation is finished and your system is fully configured, you will be asked to reset your computer. This indicates that your system has been successfully installed.

Configuring MILO

In order to boot your newly installed system, you'll need to use your MILO diskette. To boot Red Hat Linux/Alpha from MILO, you must use the `boot` command. The command differs slightly depending on where your root partition is. For example, if your root partition is the second partition on your first SCSI hard drive, you would boot as follows:

```
boot sda2:vmlinux.gz root=/dev/sda2
```

However, if your root partition is the third partition on your second IDE drive, you would use this command:

```
boot hdb3:vmlinux.gz root=/dev/hdb3
```

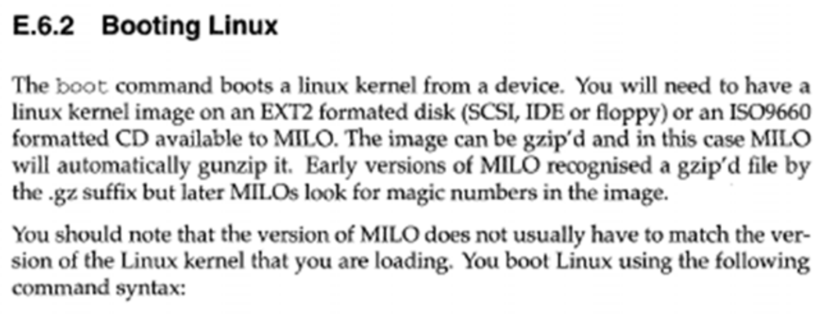
Boot your Alpha system (using the appropriate root partition name for your system, of course). Once Red Hat Linux/Alpha has finished booting, login, and issue the following command:

```
dd if-/dev/fd0 of-/dev/sda1 bs-1440k
```

This will copy MILO (along with `linload.exe`) to the small MILO partition you created during installation.

Finally, you need to create a boot selection that will look for MILO on your MILO partition. Shutdown your Alpha system, and perform the following steps from the ARC console:

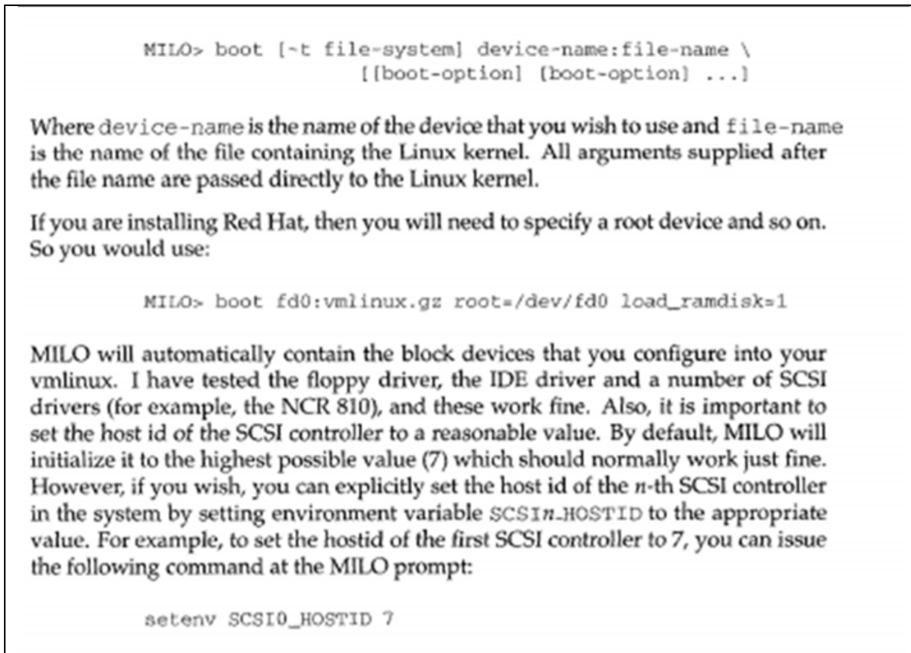
Linux Redhat, at Appendix D5.5



E.6.2 Booting Linux

The `boot` command boots a linux kernel from a device. You will need to have a linux kernel image on an EXT2 formated disk (SCSI, IDE or floppy) or an ISO9660 formatted CD available to MILO. The image can be gzip'd and in this case MILO will automatically gunzip it. Early versions of MILO recognised a gzip'd file by the .gz suffix but later MILOs look for magic numbers in the image.

You should note that the version of MILO does not usually have to match the version of the Linux kernel that you are loading. You boot Linux using the following command syntax:



```
MILO> boot [-t file-system] device-name:file-name \
                    [[boot-option] [boot-option] ...]
```

Where device-name is the name of the device that you wish to use and file-name is the name of the file containing the Linux kernel. All arguments supplied after the file name are passed directly to the Linux kernel.

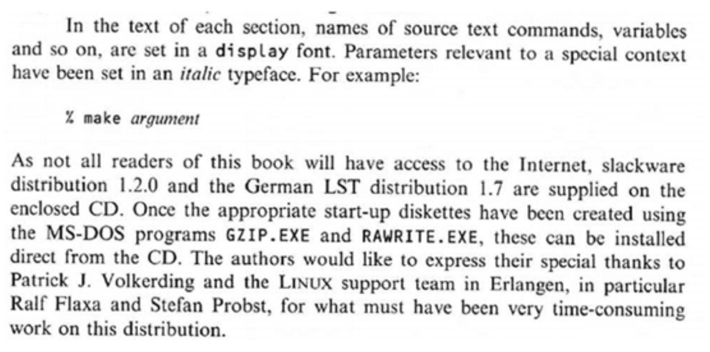
If you are installing Red Hat, then you will need to specify a root device and so on. So you would use:

```
MILO> boot fd0:vmlinux.gz root=/dev/fd0 load_ramdisk=1
```

MILO will automatically contain the block devices that you configure into your vmlinux. I have tested the floppy driver, the IDE driver and a number of SCSI drivers (for example, the NCR 810), and these work fine. Also, it is important to set the host id of the SCSI controller to a reasonable value. By default, MILO will initialize it to the highest possible value (7) which should normally work just fine. However, if you wish, you can explicitly set the host id of the *n*-th SCSI controller in the system by setting environment variable SCSI*n*_HOSTID to the appropriate value. For example, to set the hostid of the first SCSI controller to 7, you can issue the following command at the MILO prompt:

```
setenv SCSI0_HOSTID 7
```

Linux Redhat, at Appendix E.6.2

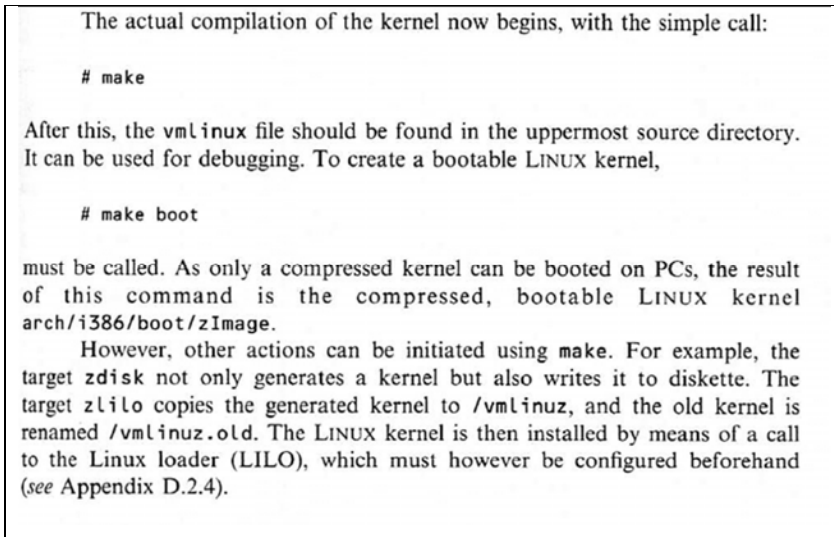


In the text of each section, names of source text commands, variables and so on, are set in a display font. Parameters relevant to a special context have been set in an *italic* typeface. For example:

```
% make argument
```

As not all readers of this book will have access to the Internet, slackware distribution 1.2.0 and the German LST distribution 1.7 are supplied on the enclosed CD. Once the appropriate start-up diskettes have been created using the MS-DOS programs GZIP.EXE and RAWRITE.EXE, these can be installed direct from the CD. The authors would like to express their special thanks to Patrick J. Volkerding and the LINUX support team in Erlangen, in particular Ralf Flaxa and Stefan Probst, for what must have been very time-consuming work on this distribution.

Beck, at Preface



The actual compilation of the kernel now begins, with the simple call:

```
# make
```

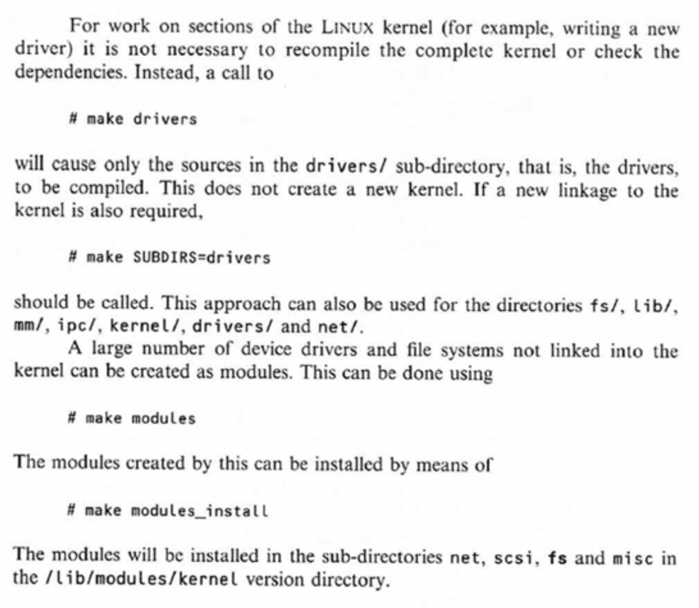
After this, the vmlinux file should be found in the uppermost source directory. It can be used for debugging. To create a bootable LINUX kernel,

```
# make boot
```

must be called. As only a compressed kernel can be booted on PCs, the result of this command is the compressed, bootable LINUX kernel arch/i386/boot/zImage.

However, other actions can be initiated using make. For example, the target zdisk not only generates a kernel but also writes it to diskette. The target zlilo copies the generated kernel to /vmlinuz, and the old kernel is renamed /vmlinuz.old. The LINUX kernel is then installed by means of a call to the Linux loader (LILO), which must however be configured beforehand (*see* Appendix D.2.4).

Beck, at 2.2



For work on sections of the LINUX kernel (for example, writing a new driver) it is not necessary to recompile the complete kernel or check the dependencies. Instead, a call to

```
# make drivers
```

will cause only the sources in the drivers/ sub-directory, that is, the drivers, to be compiled. This does not create a new kernel. If a new linkage to the kernel is also required,

```
# make SUBDIRS=drivers
```

should be called. This approach can also be used for the directories fs/, lib/, mm/, ipc/, kernel/, drivers/ and net/.

A large number of device drivers and file systems not linked into the kernel can be created as modules. This can be done using

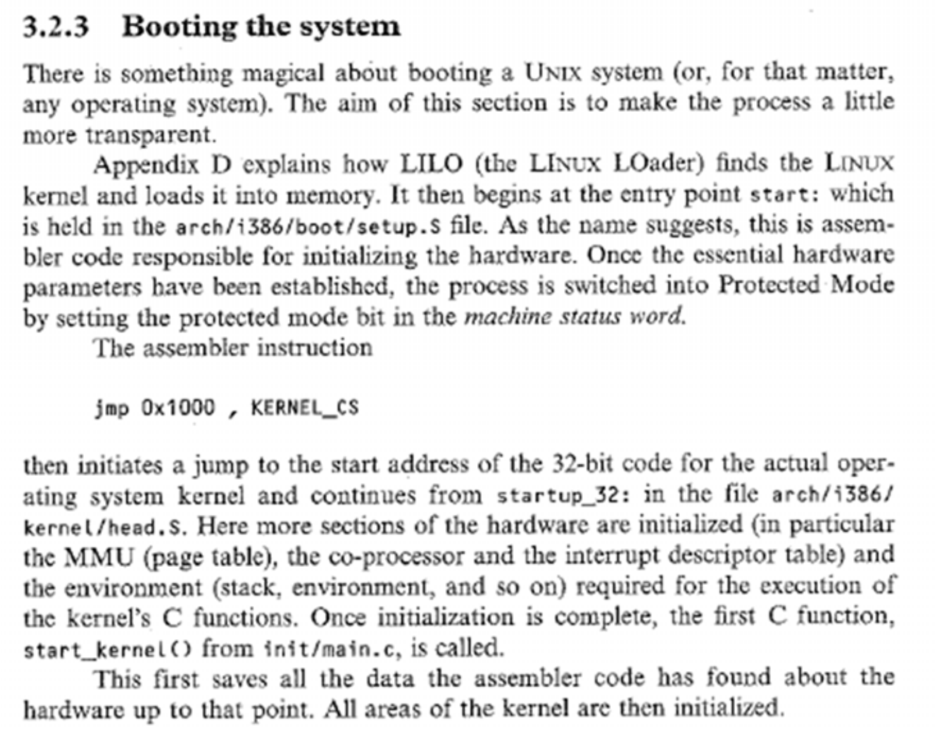
```
# make modules
```

The modules created by this can be installed by means of

```
# make modules_install
```

The modules will be installed in the sub-directories net, scsi, fs and misc in the /lib/modules/kernel version directory.

Beck, at 2.2



3.2.3 Booting the system

There is something magical about booting a UNIX system (or, for that matter, any operating system). The aim of this section is to make the process a little more transparent.

Appendix D explains how LILO (the LINUX LOader) finds the LINUX kernel and loads it into memory. It then begins at the entry point `start:` which is held in the `arch/i386/boot/setup.S` file. As the name suggests, this is assembler code responsible for initializing the hardware. Once the essential hardware parameters have been established, the process is switched into Protected Mode by setting the protected mode bit in the *machine status word*.

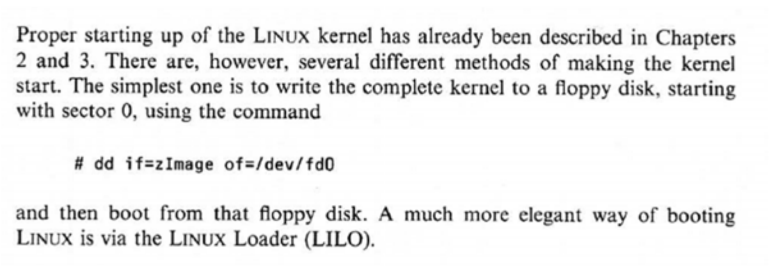
The assembler instruction

```
jmp 0x1000 , KERNEL_CS
```

then initiates a jump to the start address of the 32-bit code for the actual operating system kernel and continues from `startup_32:` in the file `arch/i386/kernel/head.S`. Here more sections of the hardware are initialized (in particular the MMU (page table), the co-processor and the interrupt descriptor table) and the environment (stack, environment, and so on) required for the execution of the kernel's C functions. Once initialization is complete, the first C function, `start_kernel()` from `init/main.c`, is called.

This first saves all the data the assembler code has found about the hardware up to that point. All areas of the kernel are then initialized.

Beck, at 3.2.3

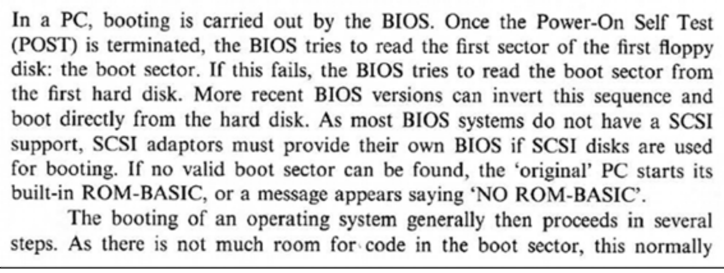


Proper starting up of the LINUX kernel has already been described in Chapters 2 and 3. There are, however, several different methods of making the kernel start. The simplest one is to write the complete kernel to a floppy disk, starting with sector 0, using the command

```
# dd if=zImage of=/dev/fd0
```

and then boot from that floppy disk. A much more elegant way of booting LINUX is via the LINUX Loader (LILO).

Beck, at Appendix D



In a PC, booting is carried out by the BIOS. Once the Power-On Self Test (POST) is terminated, the BIOS tries to read the first sector of the first floppy disk: the boot sector. If this fails, the BIOS tries to read the boot sector from the first hard disk. More recent BIOS versions can invert this sequence and boot directly from the hard disk. As most BIOS systems do not have a SCSI support, SCSI adaptors must provide their own BIOS if SCSI disks are used for booting. If no valid boot sector can be found, the 'original' PC starts its built-in ROM-BASIC, or a message appears saying 'NO ROM-BASIC'.

The booting of an operating system generally then proceeds in several steps. As there is not much room for code in the boot sector, this normally

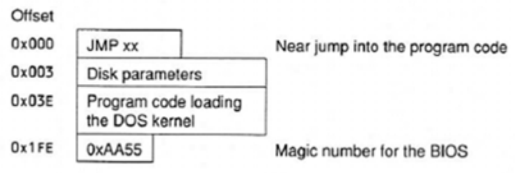

Offset
0x000
JMP xx
Near jump into the program code
0x003
Disk parameters
0x03E
Program code loading the DOS kernel
0x1FE
0xAA55
Magic number for the BIOS

Figure D.1 The MS-DOS boot sector.

loads a second loader, and so on, until the actual operating system kernel is completely loaded.

As Figure D.1 shows, the structure of a boot sector is relatively simple; its length is always 512 bytes (so that it can be stored both on a floppy disk and a hard disk).

In this, the disk parameters are significant only for MS-DOS. It is important that the code starts at offset 0 and that the boot sector is terminated by the *magic number*.

Beck, at Appendix D.1

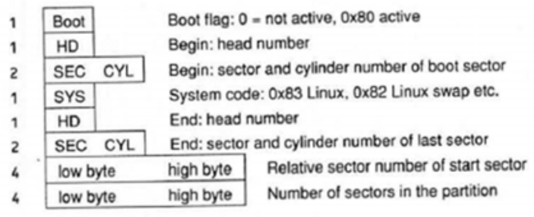

1
Boot
Boot flag: 0 = not active, 0x80 active
1
HD
Begin: head number
2
SEC CYL
Begin: sector and cylinder number of boot sector
1
SYS
System code: 0x83 Linux, 0x82 Linux swap etc.
1
HD
End: head number
2
SEC CYL
End: sector and cylinder number of last sector
4
low byte high byte
Relative sector number of start sector
4
low byte high byte
Number of sectors in the partition

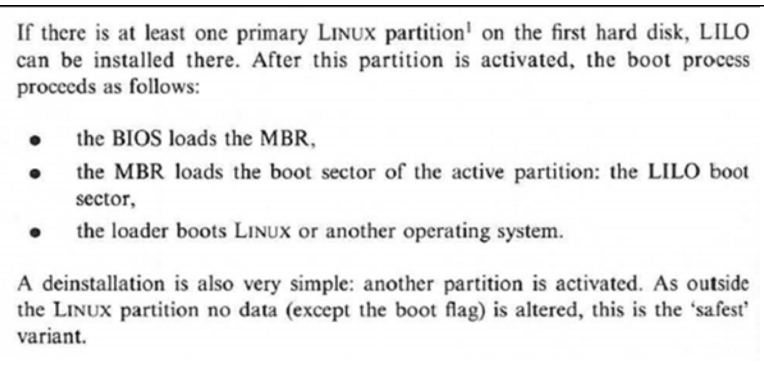
Figure D.3 Structure of a partition entry.

Originally, there were only primary partitions: therefore, `fdisk` under MS-DOS and most of the equivalent programs can only activate those partitions. The code in the MBR must thus only carry out the following operations:

- determine the active partition,
- load the boot sector of the active partition, using the BIOS,
- jump into the boot sector at offset 0.

The number of bytes in the MBR is more than sufficient to do this. Because, as described above, each partition in principle contains a boot sector, and furthermore, the structure of any second hard disk which may be present is similar to that of the first disk, a multitude of replacements for the standard MS-DOS MBR have come about: so-called *boot managers*. They all have in common the fact that they either substitute the MBR with their own code or occupy the boot sector of an active partition. To boot LINUX, most will use the LINUX loader LILO.

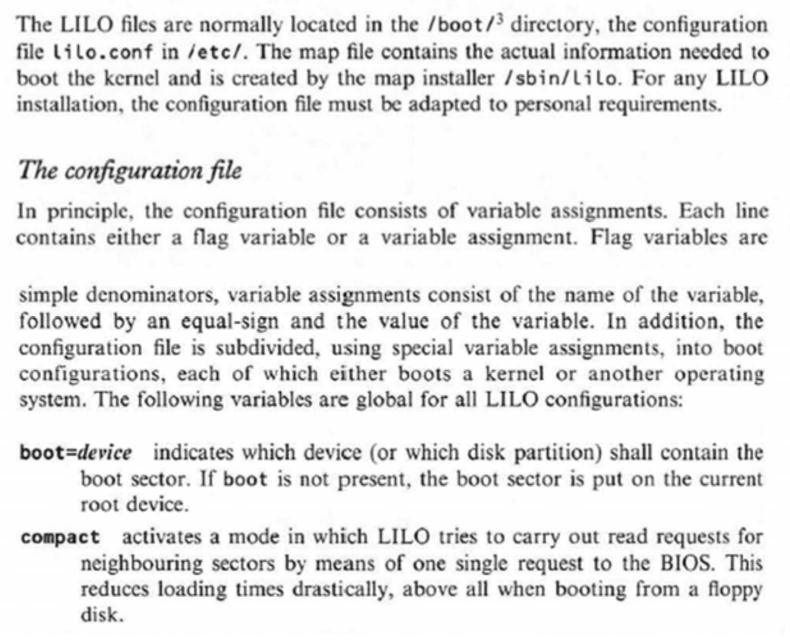
Beck, at Appendix D.1


If there is at least one primary LINUX partition[1] on the first hard disk, LILO can be installed there. After this partition is activated, the boot process proceeds as follows:

- the BIOS loads the MBR,
- the MBR loads the boot sector of the active partition: the LILO boot sector,
- the loader boots LINUX or another operating system.

A deinstallation is also very simple: another partition is activated. As outside the LINUX partition no data (except the boot flag) is altered, this is the 'safest' variant.

Beck, at Appendix D.2.1


The LILO files are normally located in the `/boot/`[3] directory, the configuration file `lilo.conf` in `/etc/`. The map file contains the actual information needed to boot the kernel and is created by the map installer `/sbin/lilo`. For any LILO installation, the configuration file must be adapted to personal requirements.

The configuration file

In principle, the configuration file consists of variable assignments. Each line contains either a flag variable or a variable assignment. Flag variables are simple denominators, variable assignments consist of the name of the variable, followed by an equal-sign and the value of the variable. In addition, the configuration file is subdivided, using special variable assignments, into boot configurations, each of which either boots a kernel or another operating system. The following variables are global for all LILO configurations:

boot=*device* indicates which device (or which disk partition) shall contain the boot sector. If **boot** is not present, the boot sector is put on the current root device.

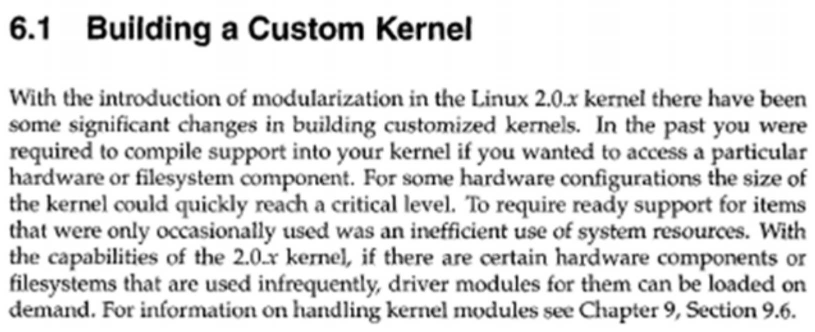
compact activates a mode in which LILO tries to carry out read requests for neighbouring sectors by means of one single request to the BIOS. This reduces loading times drastically, above all when booting from a floppy disk.

Beck, at Appendix D.2.4

1.4 accessing and decompressing said at least a portion of said boot data in said compressed form from said memory; and	Linux Kernel, as evidenced by the example citations below, discloses "accessing and decompressing said at least a portion of said boot data in said compressed form from said memory; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing and decompressing said at least a portion of said boot data in said compressed form from said memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

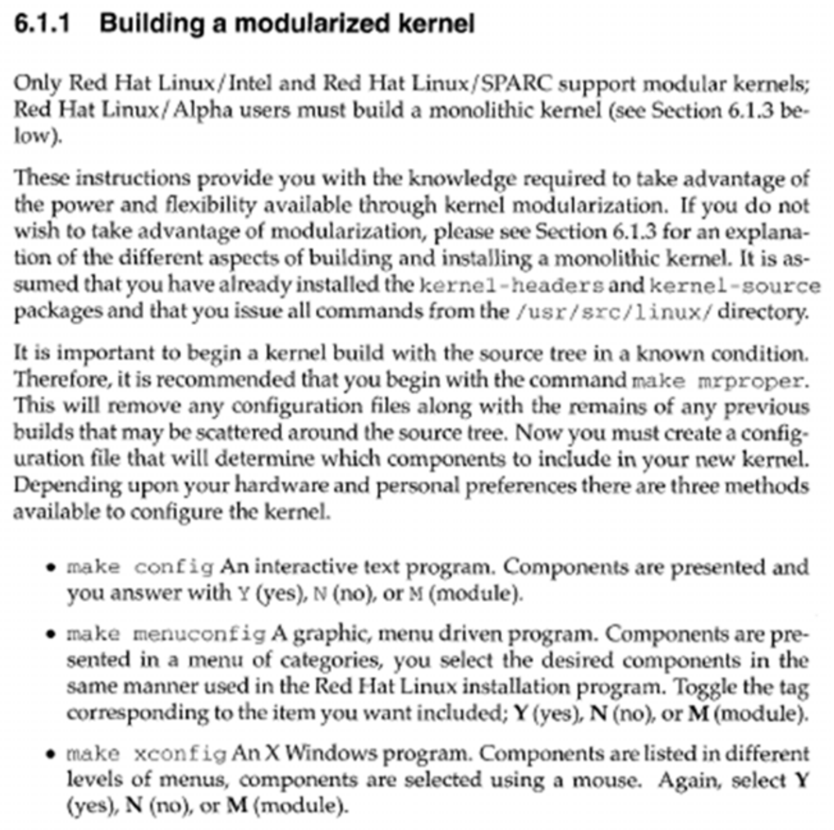
Linux Kernel discloses this limitation:



6.1 Building a Custom Kernel

With the introduction of modularization in the Linux 2.0.*x* kernel there have been some significant changes in building customized kernels. In the past you were required to compile support into your kernel if you wanted to access a particular hardware or filesystem component. For some hardware configurations the size of the kernel could quickly reach a critical level. To require ready support for items that were only occasionally used was an inefficient use of system resources. With the capabilities of the 2.0.*x* kernel, if there are certain hardware components or filesystems that are used infrequently, driver modules for them can be loaded on demand. For information on handling kernel modules see Chapter 9, Section 9.6.

Linux Redhat, at 6.1



6.1.1 Building a modularized kernel

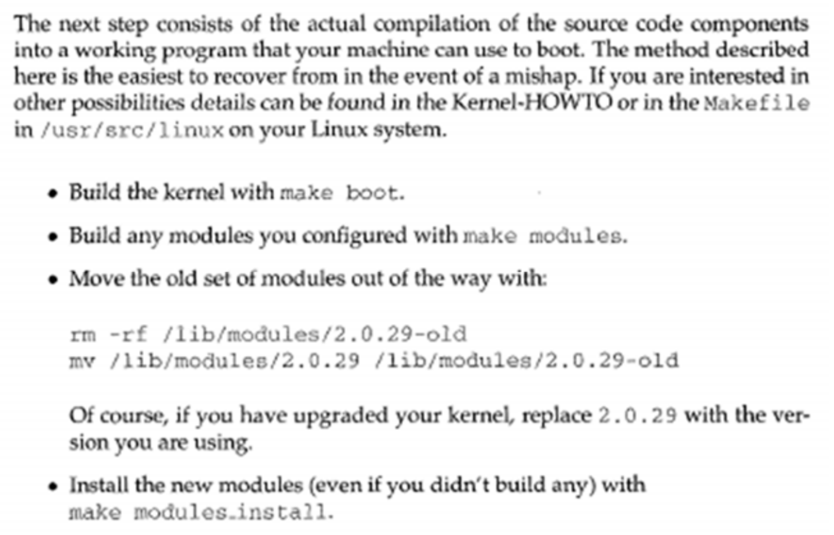
Only Red Hat Linux/Intel and Red Hat Linux/SPARC support modular kernels; Red Hat Linux/Alpha users must build a monolithic kernel (see Section 6.1.3 below).

These instructions provide you with the knowledge required to take advantage of the power and flexibility available through kernel modularization. If you do not wish to take advantage of modularization, please see Section 6.1.3 for an explanation of the different aspects of building and installing a monolithic kernel. It is assumed that you have already installed the `kernel-headers` and `kernel-source` packages and that you issue all commands from the `/usr/src/linux/` directory.

It is important to begin a kernel build with the source tree in a known condition. Therefore, it is recommended that you begin with the command `make mrproper`. This will remove any configuration files along with the remains of any previous builds that may be scattered around the source tree. Now you must create a configuration file that will determine which components to include in your new kernel. Depending upon your hardware and personal preferences there are three methods available to configure the kernel.

- `make config` An interactive text program. Components are presented and you answer with `Y` (yes), `N` (no), or `M` (module).
- `make menuconfig` A graphic, menu driven program. Components are presented in a menu of categories, you select the desired components in the same manner used in the Red Hat Linux installation program. Toggle the tag corresponding to the item you want included; **Y** (yes), **N** (no), or **M** (module).
- `make xconfig` An X Windows program. Components are listed in different levels of menus, components are selected using a mouse. Again, select **Y** (yes), **N** (no), or **M** (module).

Linux Redhat, at 6.1.1



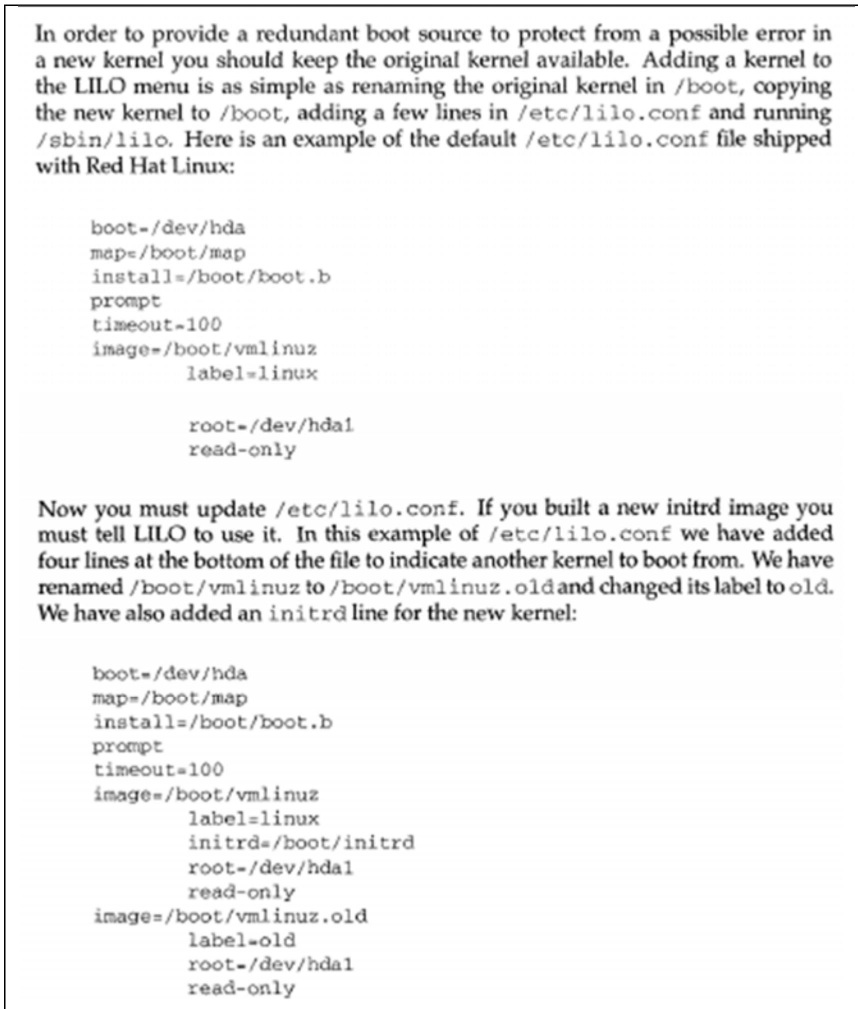
The next step consists of the actual compilation of the source code components into a working program that your machine can use to boot. The method described here is the easiest to recover from in the event of a mishap. If you are interested in other possibilities details can be found in the Kernel-HOWTO or in the `Makefile` in `/usr/src/linux` on your Linux system.

- Build the kernel with `make boot`.
- Build any modules you configured with `make modules`.
- Move the old set of modules out of the way with:

```
rm -rf /lib/modules/2.0.29-old
mv /lib/modules/2.0.29 /lib/modules/2.0.29-old
```

  Of course, if you have upgraded your kernel, replace `2.0.29` with the version you are using.
- Install the new modules (even if you didn't build any) with `make modules_install`.

Linux Redhat, at 6.1.1



In order to provide a redundant boot source to protect from a possible error in a new kernel you should keep the original kernel available. Adding a kernel to the LILO menu is as simple as renaming the original kernel in /boot, copying the new kernel to /boot, adding a few lines in /etc/lilo.conf and running /sbin/lilo. Here is an example of the default /etc/lilo.conf file shipped with Red Hat Linux:

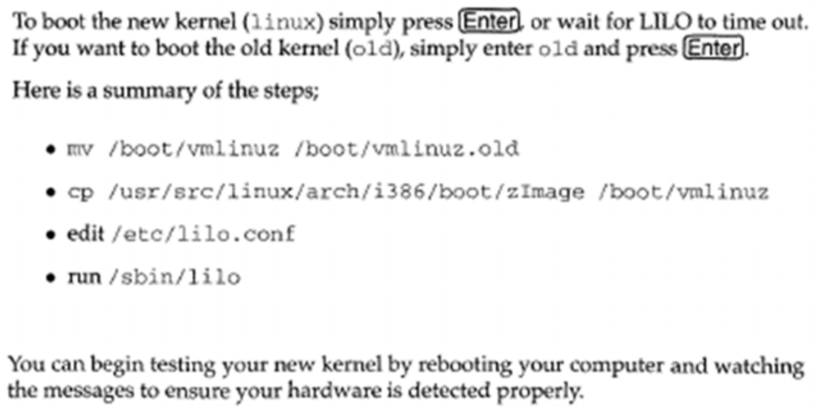
```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=100
image=/boot/vmlinuz
        label=linux

        root=/dev/hda1
        read-only
```

Now you must update /etc/lilo.conf. If you built a new initrd image you must tell LILO to use it. In this example of /etc/lilo.conf we have added four lines at the bottom of the file to indicate another kernel to boot from. We have renamed /boot/vmlinuz to /boot/vmlinuz.old and changed its label to old. We have also added an initrd line for the new kernel:

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=100
image=/boot/vmlinuz
        label=linux
        initrd=/boot/initrd
        root=/dev/hda1
        read-only
image=/boot/vmlinuz.old
        label=old
        root=/dev/hda1
        read-only
```

Linux Redhat, at 6.1.1



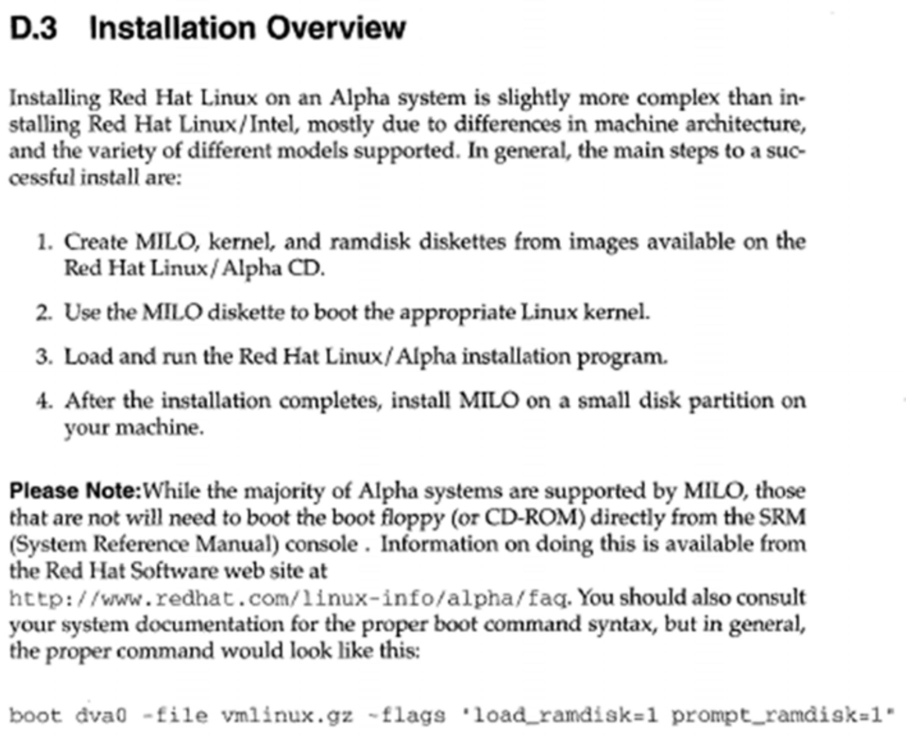
To boot the new kernel (linux) simply press Enter, or wait for LILO to time out. If you want to boot the old kernel (old), simply enter old and press Enter.

Here is a summary of the steps;

- mv /boot/vmlinuz /boot/vmlinuz.old
- cp /usr/src/linux/arch/i386/boot/zImage /boot/vmlinuz
- edit /etc/lilo.conf
- run /sbin/lilo

You can begin testing your new kernel by rebooting your computer and watching the messages to ensure your hardware is detected properly.

Linux Redhat, at 6.1.1



D.3 Installation Overview

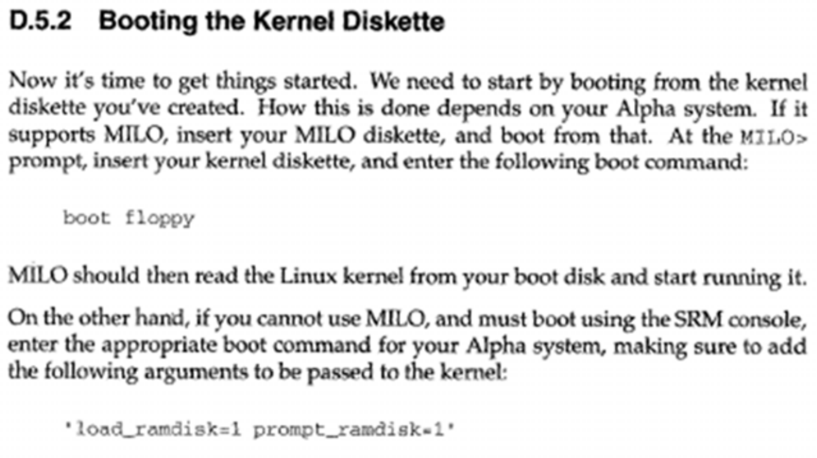
Installing Red Hat Linux on an Alpha system is slightly more complex than installing Red Hat Linux/Intel, mostly due to differences in machine architecture, and the variety of different models supported. In general, the main steps to a successful install are:

1. Create MILO, kernel, and ramdisk diskettes from images available on the Red Hat Linux/Alpha CD.
2. Use the MILO diskette to boot the appropriate Linux kernel.
3. Load and run the Red Hat Linux/Alpha installation program.
4. After the installation completes, install MILO on a small disk partition on your machine.

Please Note: While the majority of Alpha systems are supported by MILO, those that are not will need to boot the boot floppy (or CD-ROM) directly from the SRM (System Reference Manual) console . Information on doing this is available from the Red Hat Software web site at `http://www.redhat.com/linux-info/alpha/faq`. You should also consult your system documentation for the proper boot command syntax, but in general, the proper command would look like this:

```
boot dva0 -file vmlinux.gz -flags 'load_ramdisk=1 prompt_ramdisk=1"
```

Linux Redhat, at Appendix D.3



D.5.2 Booting the Kernel Diskette

Now it's time to get things started. We need to start by booting from the kernel diskette you've created. How this is done depends on your Alpha system. If it supports MILO, insert your MILO diskette, and boot from that. At the `MILO>` prompt, insert your kernel diskette, and enter the following boot command:

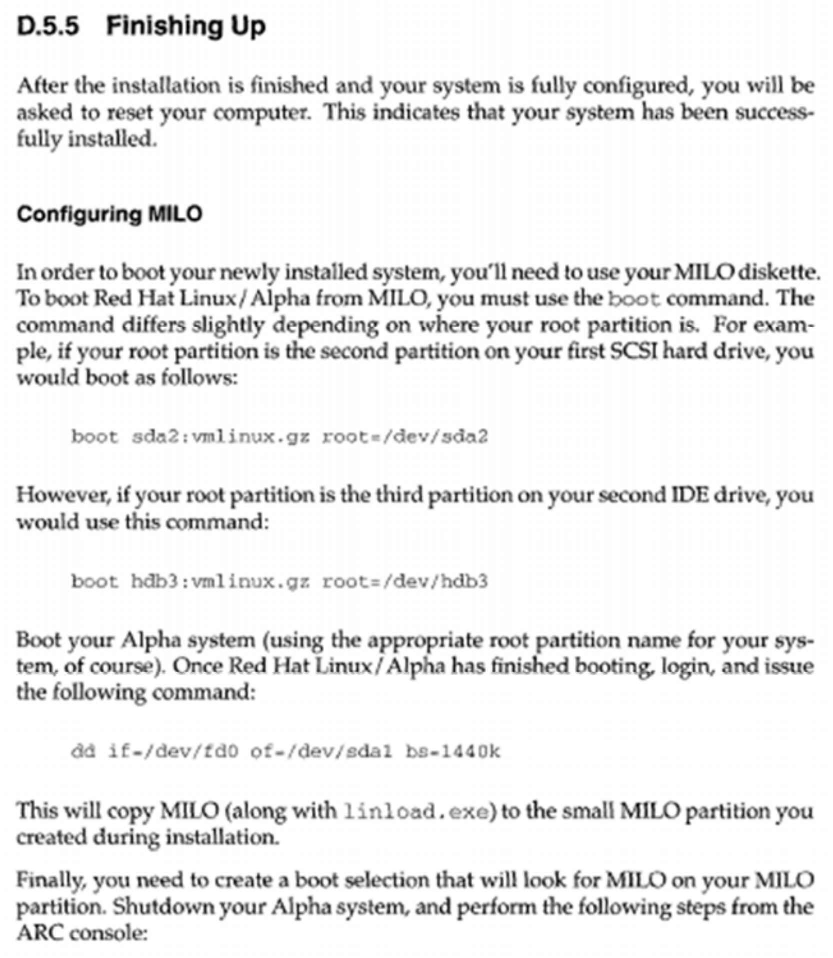
```
boot floppy
```

MILO should then read the Linux kernel from your boot disk and start running it.

On the other hand, if you cannot use MILO, and must boot using the SRM console, enter the appropriate boot command for your Alpha system, making sure to add the following arguments to be passed to the kernel:

```
'load_ramdisk=1 prompt_ramdisk=1'
```

Linux Redhat, at Appendix D.5.2



D.5.5 Finishing Up

After the installation is finished and your system is fully configured, you will be asked to reset your computer. This indicates that your system has been successfully installed.

Configuring MILO

In order to boot your newly installed system, you'll need to use your MILO diskette. To boot Red Hat Linux/Alpha from MILO, you must use the `boot` command. The command differs slightly depending on where your root partition is. For example, if your root partition is the second partition on your first SCSI hard drive, you would boot as follows:

```
boot sda2:vmlinux.gz root=/dev/sda2
```

However, if your root partition is the third partition on your second IDE drive, you would use this command:

```
boot hdb3:vmlinux.gz root=/dev/hdb3
```

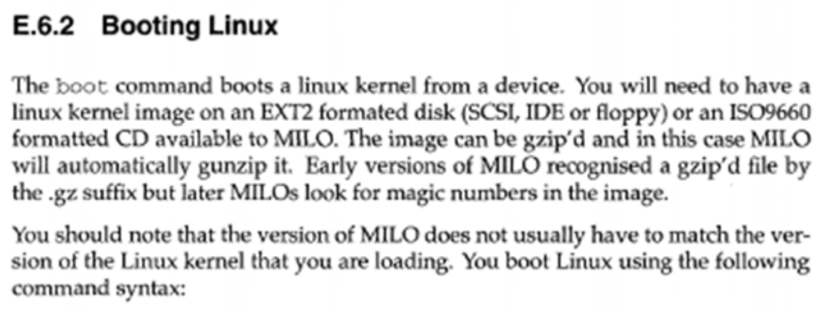
Boot your Alpha system (using the appropriate root partition name for your system, of course). Once Red Hat Linux/Alpha has finished booting, login, and issue the following command:

```
dd if=/dev/fd0 of=/dev/sda1 bs=1440k
```

This will copy MILO (along with `linload.exe`) to the small MILO partition you created during installation.

Finally, you need to create a boot selection that will look for MILO on your MILO partition. Shutdown your Alpha system, and perform the following steps from the ARC console:

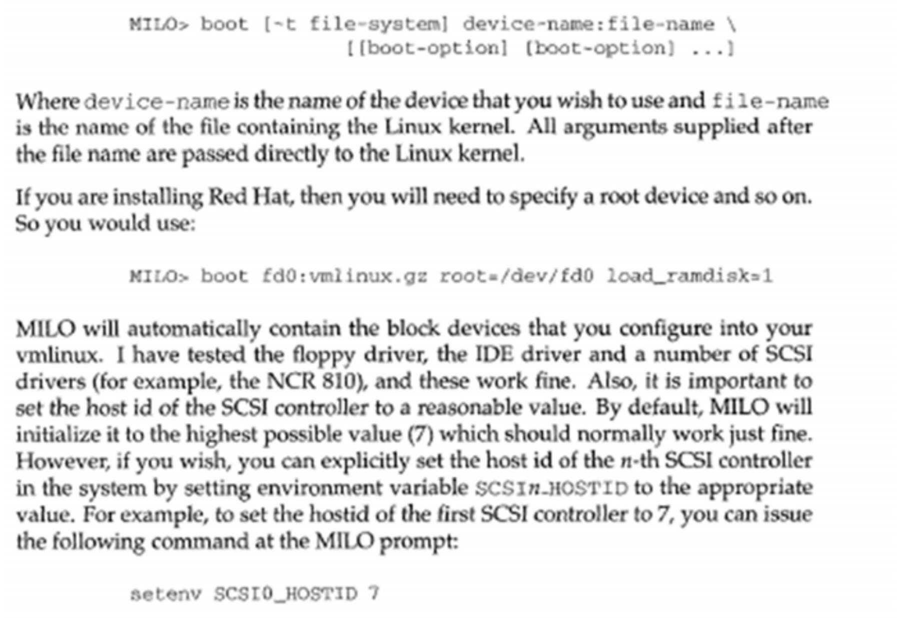
Linux Redhat, at Appendix D5.5



E.6.2 Booting Linux

The `boot` command boots a linux kernel from a device. You will need to have a linux kernel image on an EXT2 formated disk (SCSI, IDE or floppy) or an ISO9660 formatted CD available to MILO. The image can be gzip'd and in this case MILO will automatically gunzip it. Early versions of MILO recognised a gzip'd file by the .gz suffix but later MILOs look for magic numbers in the image.

You should note that the version of MILO does not usually have to match the version of the Linux kernel that you are loading. You boot Linux using the following command syntax:



```
MILO> boot [-t file-system] device-name:file-name \
                  [[boot-option] [boot-option] ...]
```

Where `device-name` is the name of the device that you wish to use and `file-name` is the name of the file containing the Linux kernel. All arguments supplied after the file name are passed directly to the Linux kernel.

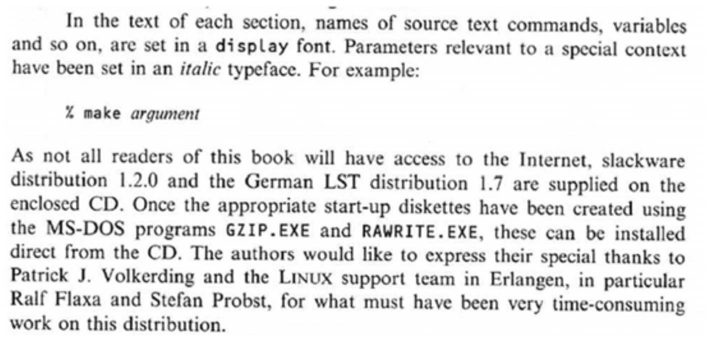
If you are installing Red Hat, then you will need to specify a root device and so on. So you would use:

```
MILO> boot fd0:vmlinux.gz root=/dev/fd0 load_ramdisk=1
```

MILO will automatically contain the block devices that you configure into your vmlinux. I have tested the floppy driver, the IDE driver and a number of SCSI drivers (for example, the NCR 810), and these work fine. Also, it is important to set the host id of the SCSI controller to a reasonable value. By default, MILO will initialize it to the highest possible value (7) which should normally work just fine. However, if you wish, you can explicitly set the host id of the *n*-th SCSI controller in the system by setting environment variable `SCSIn_HOSTID` to the appropriate value. For example, to set the hostid of the first SCSI controller to 7, you can issue the following command at the MILO prompt:

```
setenv SCSI0_HOSTID 7
```

Linux Redhat, at Appendix E.6.2

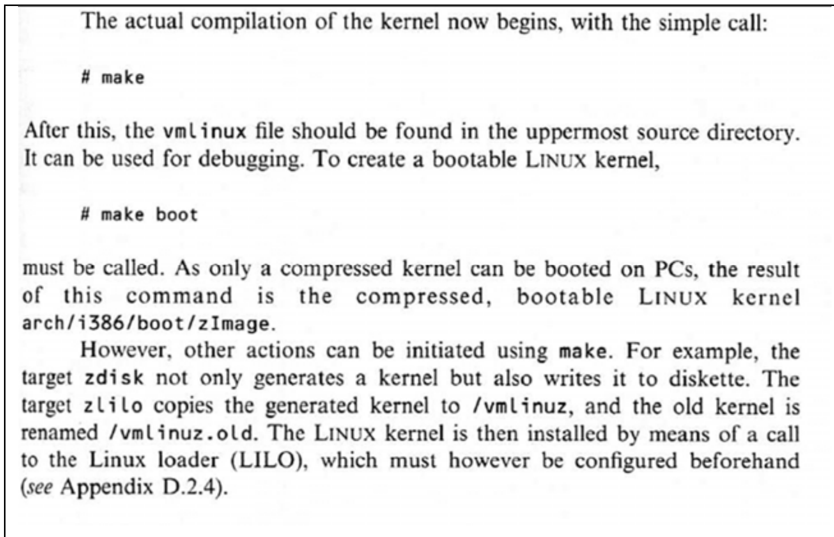


In the text of each section, names of source text commands, variables and so on, are set in a `display` font. Parameters relevant to a special context have been set in an *italic* typeface. For example:

```
% make argument
```

As not all readers of this book will have access to the Internet, slackware distribution 1.2.0 and the German LST distribution 1.7 are supplied on the enclosed CD. Once the appropriate start-up diskettes have been created using the MS-DOS programs `GZIP.EXE` and `RAWRITE.EXE`, these can be installed direct from the CD. The authors would like to express their special thanks to Patrick J. Volkerding and the LINUX support team in Erlangen, in particular Ralf Flaxa and Stefan Probst, for what must have been very time-consuming work on this distribution.

Beck, at Preface


The actual compilation of the kernel now begins, with the simple call:

```
# make
```

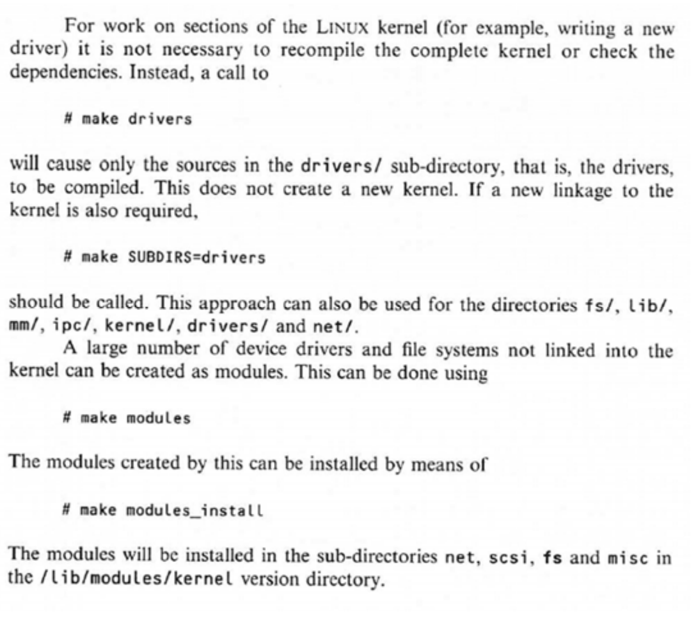
After this, the vmlinux file should be found in the uppermost source directory. It can be used for debugging. To create a bootable LINUX kernel,

```
# make boot
```

must be called. As only a compressed kernel can be booted on PCs, the result of this command is the compressed, bootable LINUX kernel arch/i386/boot/zImage.

However, other actions can be initiated using make. For example, the target zdisk not only generates a kernel but also writes it to diskette. The target zlilo copies the generated kernel to /vmlinuz, and the old kernel is renamed /vmlinuz.old. The LINUX kernel is then installed by means of a call to the Linux loader (LILO), which must however be configured beforehand (*see* Appendix D.2.4).

Beck, at 2.2


For work on sections of the LINUX kernel (for example, writing a new driver) it is not necessary to recompile the complete kernel or check the dependencies. Instead, a call to

```
# make drivers
```

will cause only the sources in the drivers/ sub-directory, that is, the drivers, to be compiled. This does not create a new kernel. If a new linkage to the kernel is also required,

```
# make SUBDIRS=drivers
```

should be called. This approach can also be used for the directories fs/, lib/, mm/, ipc/, kernel/, drivers/ and net/.

A large number of device drivers and file systems not linked into the kernel can be created as modules. This can be done using

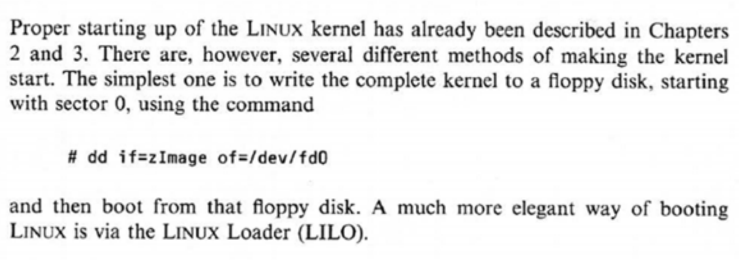
```
# make modules
```

The modules created by this can be installed by means of

```
# make modules_install
```

The modules will be installed in the sub-directories net, scsi, fs and misc in the /lib/modules/kernel version directory.

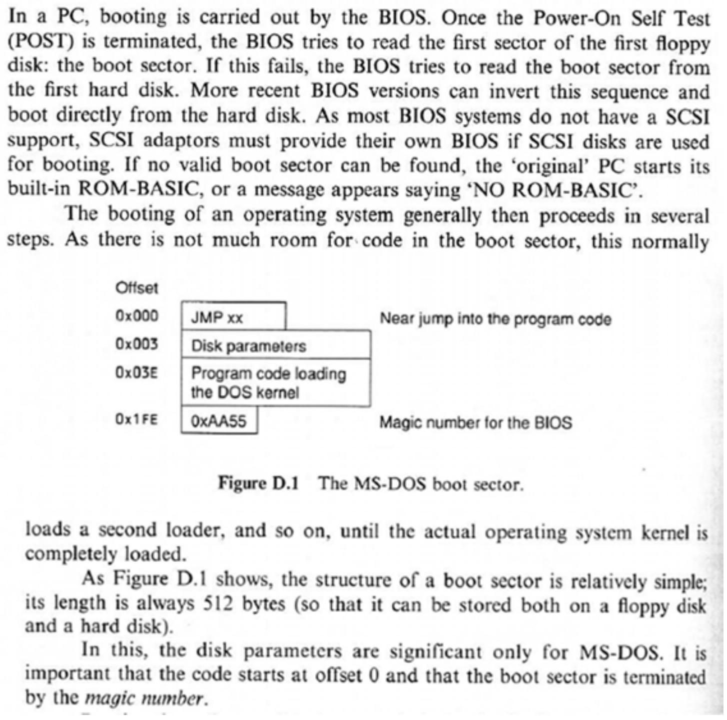
Beck, at 2.2


Proper starting up of the LINUX kernel has already been described in Chapters 2 and 3. There are, however, several different methods of making the kernel start. The simplest one is to write the complete kernel to a floppy disk, starting with sector 0, using the command

```
# dd if=zImage of=/dev/fd0
```

and then boot from that floppy disk. A much more elegant way of booting LINUX is via the LINUX Loader (LILO).

Beck, at Appendix D



In a PC, booting is carried out by the BIOS. Once the Power-On Self Test (POST) is terminated, the BIOS tries to read the first sector of the first floppy disk: the boot sector. If this fails, the BIOS tries to read the boot sector from the first hard disk. More recent BIOS versions can invert this sequence and boot directly from the hard disk. As most BIOS systems do not have a SCSI support, SCSI adaptors must provide their own BIOS if SCSI disks are used for booting. If no valid boot sector can be found, the 'original' PC starts its built-in ROM-BASIC, or a message appears saying 'NO ROM-BASIC'.

The booting of an operating system generally then proceeds in several steps. As there is not much room for code in the boot sector, this normally

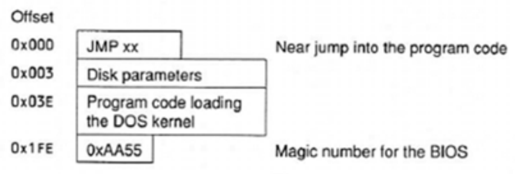

Offset
0x000
JMP xx
Near jump into the program code
0x003
Disk parameters
0x03E
Program code loading the DOS kernel
0x1FE
0xAA55
Magic number for the BIOS

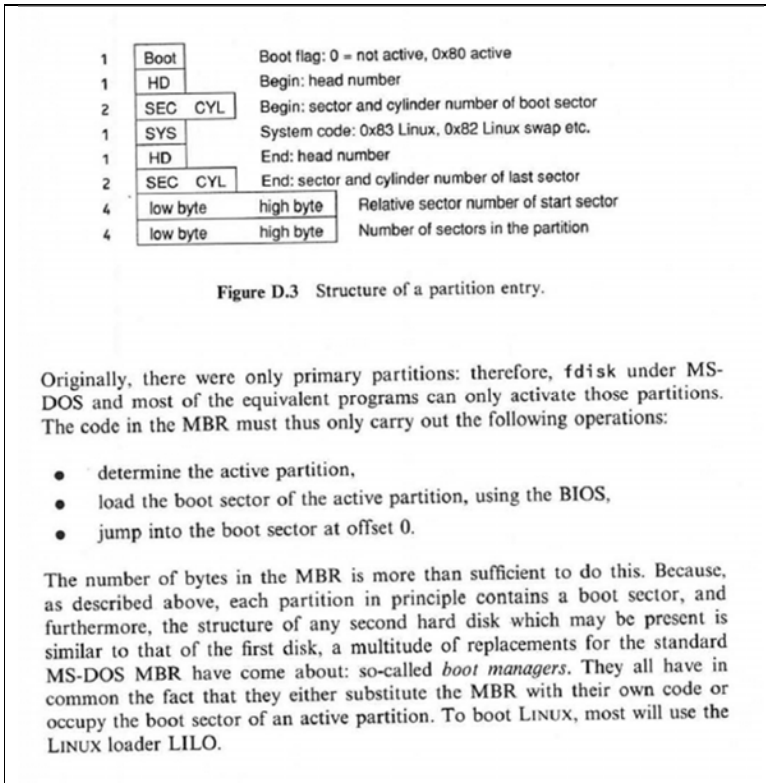
Figure D.1 The MS-DOS boot sector.

loads a second loader, and so on, until the actual operating system kernel is completely loaded.

As Figure D.1 shows, the structure of a boot sector is relatively simple; its length is always 512 bytes (so that it can be stored both on a floppy disk and a hard disk).

In this, the disk parameters are significant only for MS-DOS. It is important that the code starts at offset 0 and that the boot sector is terminated by the *magic number*.

Beck, at Appendix D.1



1	Boot	Boot flag: 0 = not active, 0x80 active
1	HD	Begin: head number
2	SEC CYL	Begin: sector and cylinder number of boot sector
1	SYS	System code: 0x83 Linux, 0x82 Linux swap etc.
1	HD	End: head number
2	SEC CYL	End: sector and cylinder number of last sector
4	low byte high byte	Relative sector number of start sector
4	low byte high byte	Number of sectors in the partition

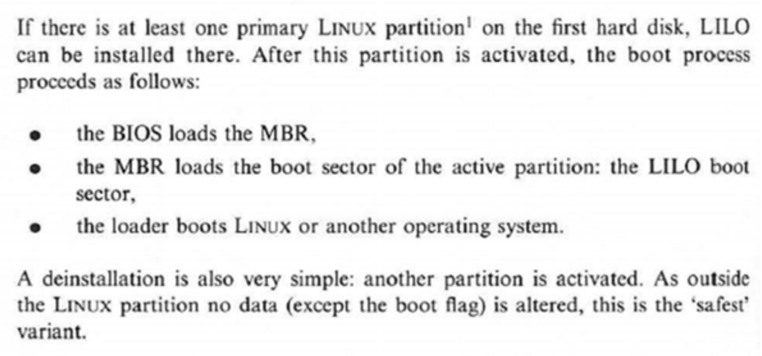
Figure D.3 Structure of a partition entry.

Originally, there were only primary partitions: therefore, `fdisk` under MS-DOS and most of the equivalent programs can only activate those partitions. The code in the MBR must thus only carry out the following operations:

- determine the active partition,
- load the boot sector of the active partition, using the BIOS,
- jump into the boot sector at offset 0.

The number of bytes in the MBR is more than sufficient to do this. Because, as described above, each partition in principle contains a boot sector, and furthermore, the structure of any second hard disk which may be present is similar to that of the first disk, a multitude of replacements for the standard MS-DOS MBR have come about: so-called *boot managers*. They all have in common the fact that they either substitute the MBR with their own code or occupy the boot sector of an active partition. To boot LINUX, most will use the LINUX loader LILO.

Beck, at Appendix D.1

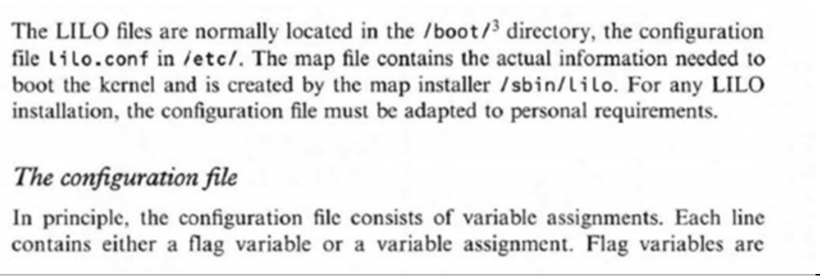


If there is at least one primary LINUX partition[1] on the first hard disk, LILO can be installed there. After this partition is activated, the boot process proceeds as follows:

- the BIOS loads the MBR,
- the MBR loads the boot sector of the active partition: the LILO boot sector,
- the loader boots LINUX or another operating system.

A deinstallation is also very simple: another partition is activated. As outside the LINUX partition no data (except the boot flag) is altered, this is the 'safest' variant.

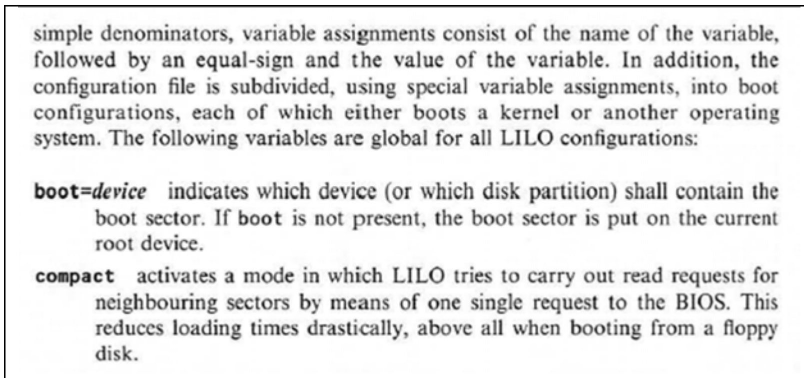
Beck, at Appendix D.2.1



The LILO files are normally located in the `/boot/`[3] directory, the configuration file `lilo.conf` in `/etc/`. The map file contains the actual information needed to boot the kernel and is created by the map installer `/sbin/lilo`. For any LILO installation, the configuration file must be adapted to personal requirements.

The configuration file

In principle, the configuration file consists of variable assignments. Each line contains either a flag variable or a variable assignment. Flag variables are


simple denominators, variable assignments consist of the name of the variable, followed by an equal-sign and the value of the variable. In addition, the configuration file is subdivided, using special variable assignments, into boot configurations, each of which either boots a kernel or another operating system. The following variables are global for all LILO configurations:

boot=*device* indicates which device (or which disk partition) shall contain the boot sector. If **boot** is not present, the boot sector is put on the current root device.

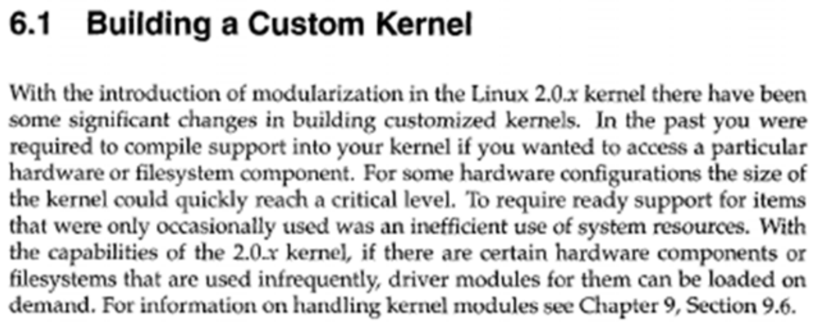
compact activates a mode in which LILO tries to carry out read requests for neighbouring sectors by means of one single request to the BIOS. This reduces loading times drastically, above all when booting from a floppy disk.

Beck, at Appendix D.2.4

1.5 utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine.	Linux Kernel, as evidenced by the example citations below, discloses "utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

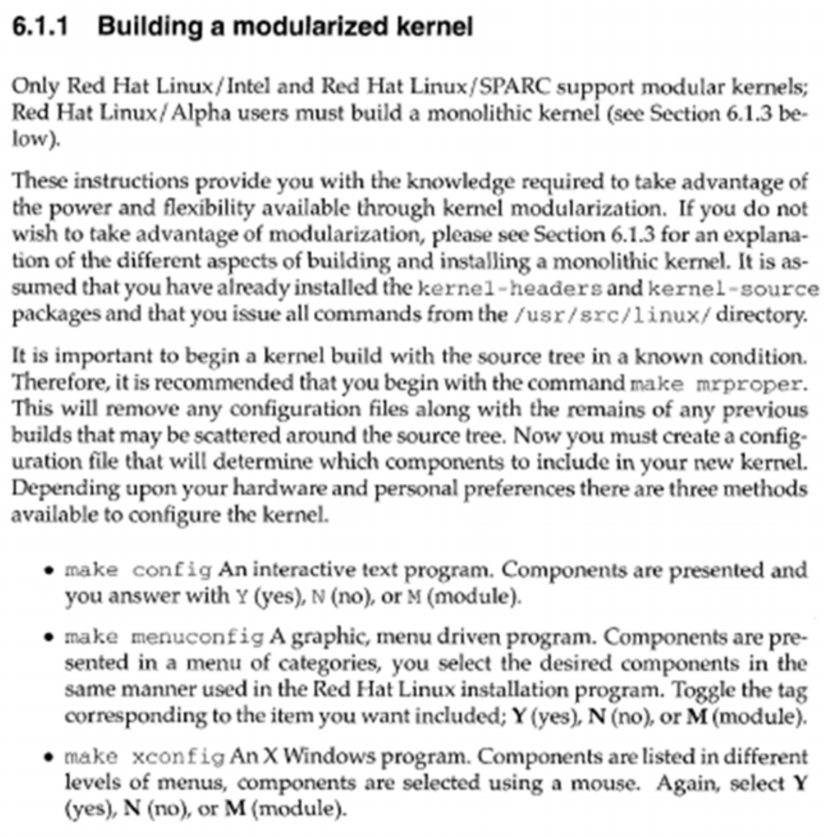
Linux Kernel discloses this limitation:



6.1 Building a Custom Kernel

With the introduction of modularization in the Linux 2.0.*x* kernel there have been some significant changes in building customized kernels. In the past you were required to compile support into your kernel if you wanted to access a particular hardware or filesystem component. For some hardware configurations the size of the kernel could quickly reach a critical level. To require ready support for items that were only occasionally used was an inefficient use of system resources. With the capabilities of the 2.0.*x* kernel, if there are certain hardware components or filesystems that are used infrequently, driver modules for them can be loaded on demand. For information on handling kernel modules see Chapter 9, Section 9.6.

Linux Redhat, at 6.1



6.1.1 Building a modularized kernel

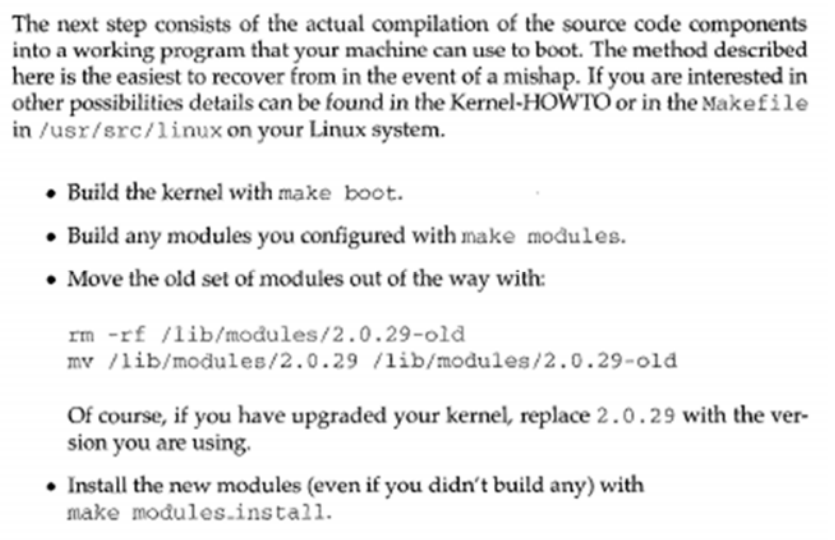
Only Red Hat Linux/Intel and Red Hat Linux/SPARC support modular kernels; Red Hat Linux/Alpha users must build a monolithic kernel (see Section 6.1.3 below).

These instructions provide you with the knowledge required to take advantage of the power and flexibility available through kernel modularization. If you do not wish to take advantage of modularization, please see Section 6.1.3 for an explanation of the different aspects of building and installing a monolithic kernel. It is assumed that you have already installed the `kernel-headers` and `kernel-source` packages and that you issue all commands from the `/usr/src/linux/` directory.

It is important to begin a kernel build with the source tree in a known condition. Therefore, it is recommended that you begin with the command `make mrproper`. This will remove any configuration files along with the remains of any previous builds that may be scattered around the source tree. Now you must create a configuration file that will determine which components to include in your new kernel. Depending upon your hardware and personal preferences there are three methods available to configure the kernel.

- `make config` An interactive text program. Components are presented and you answer with `Y` (yes), `N` (no), or `M` (module).
- `make menuconfig` A graphic, menu driven program. Components are presented in a menu of categories, you select the desired components in the same manner used in the Red Hat Linux installation program. Toggle the tag corresponding to the item you want included; **Y** (yes), **N** (no), or **M** (module).
- `make xconfig` An X Windows program. Components are listed in different levels of menus, components are selected using a mouse. Again, select **Y** (yes), **N** (no), or **M** (module).

Linux Redhat, at 6.1.1



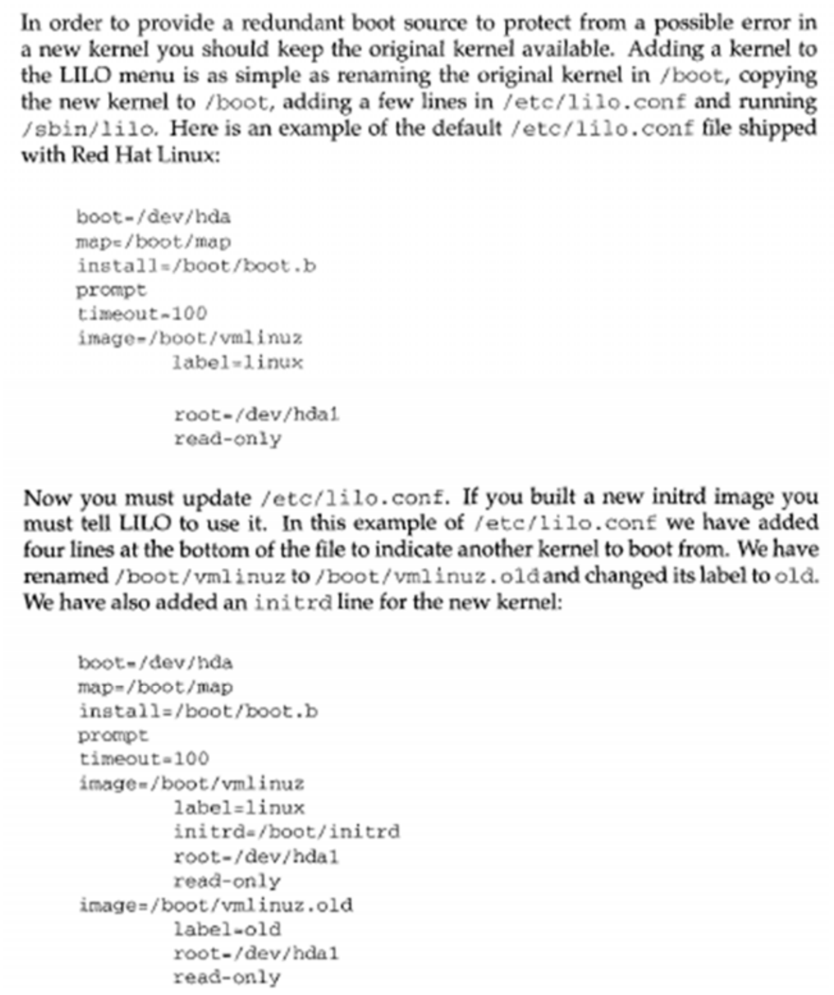
The next step consists of the actual compilation of the source code components into a working program that your machine can use to boot. The method described here is the easiest to recover from in the event of a mishap. If you are interested in other possibilities details can be found in the Kernel-HOWTO or in the `Makefile` in `/usr/src/linux` on your Linux system.

- Build the kernel with `make boot`.
- Build any modules you configured with `make modules`.
- Move the old set of modules out of the way with:

```
rm -rf /lib/modules/2.0.29-old
mv /lib/modules/2.0.29 /lib/modules/2.0.29-old
```

  Of course, if you have upgraded your kernel, replace `2.0.29` with the version you are using.
- Install the new modules (even if you didn't build any) with `make modules_install`.

Linux Redhat, at 6.1.1



In order to provide a redundant boot source to protect from a possible error in a new kernel you should keep the original kernel available. Adding a kernel to the LILO menu is as simple as renaming the original kernel in `/boot`, copying the new kernel to `/boot`, adding a few lines in `/etc/lilo.conf` and running `/sbin/lilo`. Here is an example of the default `/etc/lilo.conf` file shipped with Red Hat Linux:

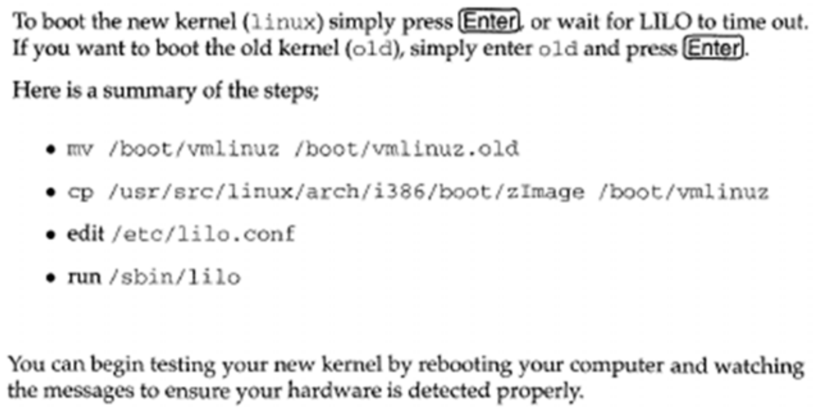
```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=100
image=/boot/vmlinuz
        label=linux

        root=/dev/hda1
        read-only
```

Now you must update `/etc/lilo.conf`. If you built a new initrd image you must tell LILO to use it. In this example of `/etc/lilo.conf` we have added four lines at the bottom of the file to indicate another kernel to boot from. We have renamed `/boot/vmlinuz` to `/boot/vmlinuz.old` and changed its label to `old`. We have also added an `initrd` line for the new kernel:

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=100
image=/boot/vmlinuz
        label=linux
        initrd=/boot/initrd
        root=/dev/hda1
        read-only
image=/boot/vmlinuz.old
        label=old
        root=/dev/hda1
        read-only
```

Linux Redhat, at 6.1.1



To boot the new kernel (`linux`) simply press Enter, or wait for LILO to time out. If you want to boot the old kernel (`old`), simply enter `old` and press Enter.

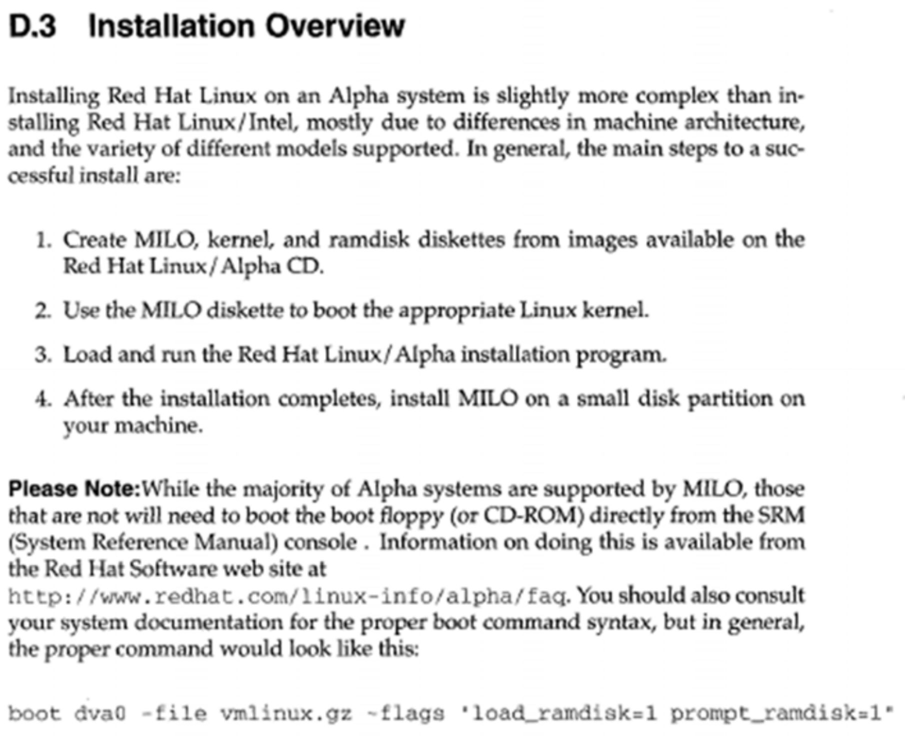
Here is a summary of the steps;

- `mv /boot/vmlinuz /boot/vmlinuz.old`
- `cp /usr/src/linux/arch/i386/boot/zImage /boot/vmlinuz`
- edit `/etc/lilo.conf`
- run `/sbin/lilo`

You can begin testing your new kernel by rebooting your computer and watching the messages to ensure your hardware is detected properly.

Linux Redhat, at 6.1.1

"utilizing said decompressed at least a portion of said boot data to boot said computer system, wherein said at least a portion of said boot data is decompressed by said data compression engine."



D.3 Installation Overview

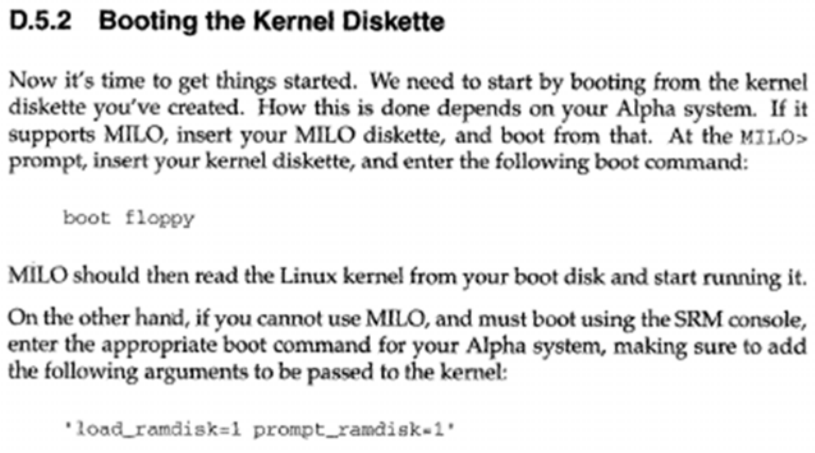
Installing Red Hat Linux on an Alpha system is slightly more complex than installing Red Hat Linux/Intel, mostly due to differences in machine architecture, and the variety of different models supported. In general, the main steps to a successful install are:

1. Create MILO, kernel, and ramdisk diskettes from images available on the Red Hat Linux/Alpha CD.
2. Use the MILO diskette to boot the appropriate Linux kernel.
3. Load and run the Red Hat Linux/Alpha installation program.
4. After the installation completes, install MILO on a small disk partition on your machine.

Please Note: While the majority of Alpha systems are supported by MILO, those that are not will need to boot the boot floppy (or CD-ROM) directly from the SRM (System Reference Manual) console . Information on doing this is available from the Red Hat Software web site at `http://www.redhat.com/linux-info/alpha/faq`. You should also consult your system documentation for the proper boot command syntax, but in general, the proper command would look like this:

```
boot dva0 -file vmlinux.gz -flags 'load_ramdisk=1 prompt_ramdisk=1'
```

Linux Redhat, at Appendix D.3



D.5.2 Booting the Kernel Diskette

Now it's time to get things started. We need to start by booting from the kernel diskette you've created. How this is done depends on your Alpha system. If it supports MILO, insert your MILO diskette, and boot from that. At the `MILO>` prompt, insert your kernel diskette, and enter the following boot command:

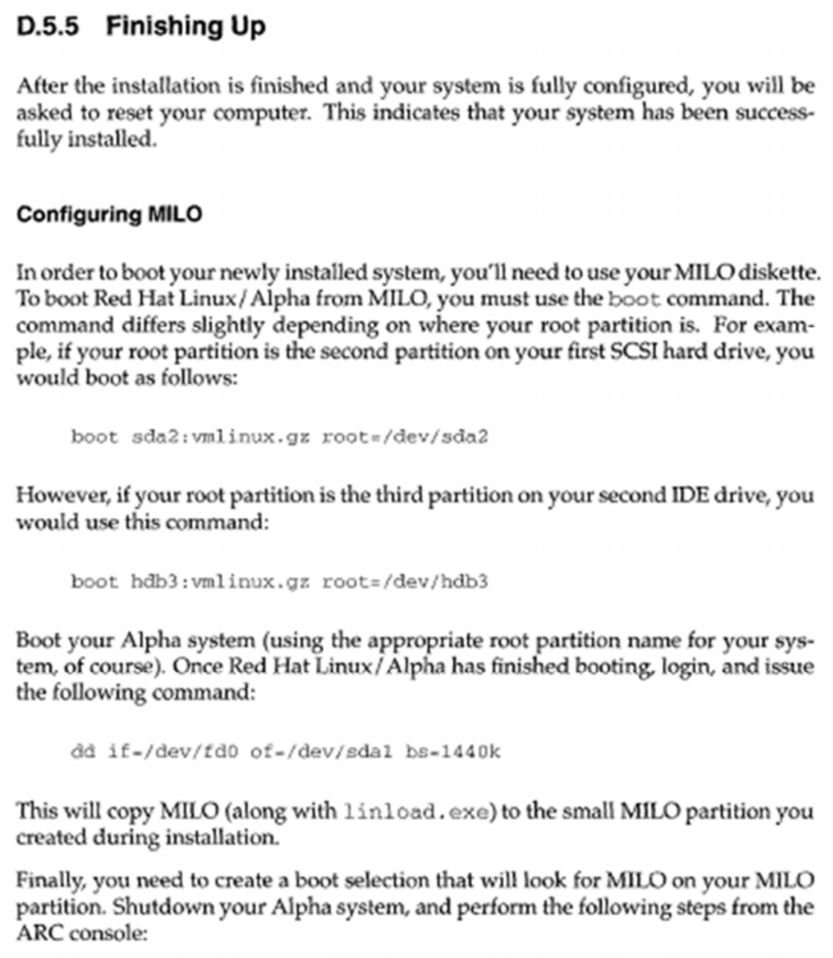
```
     boot floppy
```

MILO should then read the Linux kernel from your boot disk and start running it.

On the other hand, if you cannot use MILO, and must boot using the SRM console, enter the appropriate boot command for your Alpha system, making sure to add the following arguments to be passed to the kernel:

```
     'load_ramdisk=1 prompt_ramdisk=1'
```

Linux Redhat, at Appendix D.5.2



D.5.5 Finishing Up

After the installation is finished and your system is fully configured, you will be asked to reset your computer. This indicates that your system has been successfully installed.

Configuring MILO

In order to boot your newly installed system, you'll need to use your MILO diskette. To boot Red Hat Linux/Alpha from MILO, you must use the `boot` command. The command differs slightly depending on where your root partition is. For example, if your root partition is the second partition on your first SCSI hard drive, you would boot as follows:

```
boot sda2:vmlinux.gz root=/dev/sda2
```

However, if your root partition is the third partition on your second IDE drive, you would use this command:

```
boot hdb3:vmlinux.gz root=/dev/hdb3
```

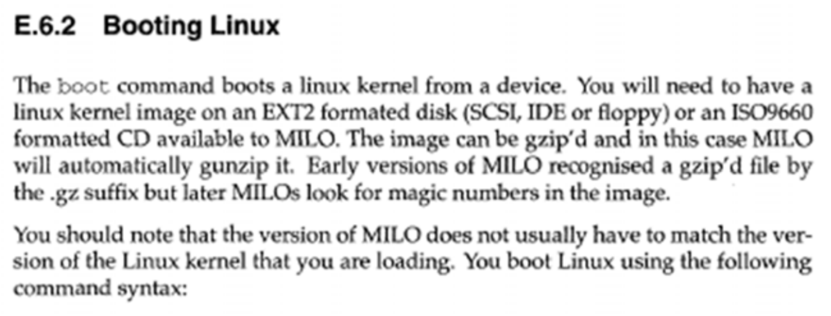
Boot your Alpha system (using the appropriate root partition name for your system, of course). Once Red Hat Linux/Alpha has finished booting, login, and issue the following command:

```
dd if=/dev/fd0 of=/dev/sda1 bs=1440k
```

This will copy MILO (along with `linload.exe`) to the small MILO partition you created during installation.

Finally, you need to create a boot selection that will look for MILO on your MILO partition. Shutdown your Alpha system, and perform the following steps from the ARC console:

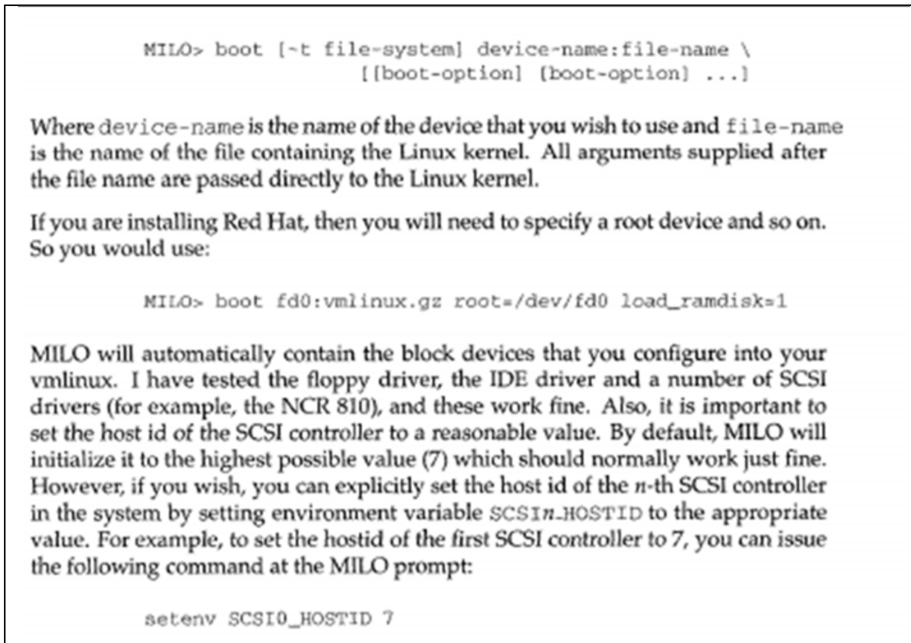
Linux Redhat, at Appendix D5.5



E.6.2 Booting Linux

The `boot` command boots a linux kernel from a device. You will need to have a linux kernel image on an EXT2 formated disk (SCSI, IDE or floppy) or an ISO9660 formatted CD available to MILO. The image can be gzip'd and in this case MILO will automatically gunzip it. Early versions of MILO recognised a gzip'd file by the .gz suffix but later MILOs look for magic numbers in the image.

You should note that the version of MILO does not usually have to match the version of the Linux kernel that you are loading. You boot Linux using the following command syntax:



```
MILO> boot [-t file-system] device-name:file-name \
                    [[boot-option] [boot-option] ...]
```

Where device-name is the name of the device that you wish to use and file-name is the name of the file containing the Linux kernel. All arguments supplied after the file name are passed directly to the Linux kernel.

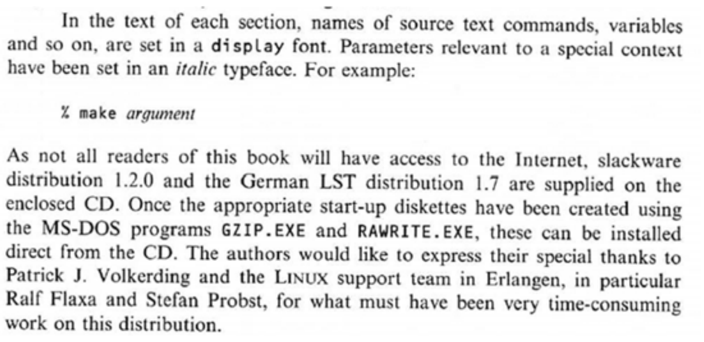
If you are installing Red Hat, then you will need to specify a root device and so on. So you would use:

```
MILO> boot fd0:vmlinux.gz root=/dev/fd0 load_ramdisk=1
```

MILO will automatically contain the block devices that you configure into your vmlinux. I have tested the floppy driver, the IDE driver and a number of SCSI drivers (for example, the NCR 810), and these work fine. Also, it is important to set the host id of the SCSI controller to a reasonable value. By default, MILO will initialize it to the highest possible value (7) which should normally work just fine. However, if you wish, you can explicitly set the host id of the *n*-th SCSI controller in the system by setting environment variable SCSI*n*_HOSTID to the appropriate value. For example, to set the hostid of the first SCSI controller to 7, you can issue the following command at the MILO prompt:

```
setenv SCSI0_HOSTID 7
```

Linux Redhat, at Appendix E.6.2

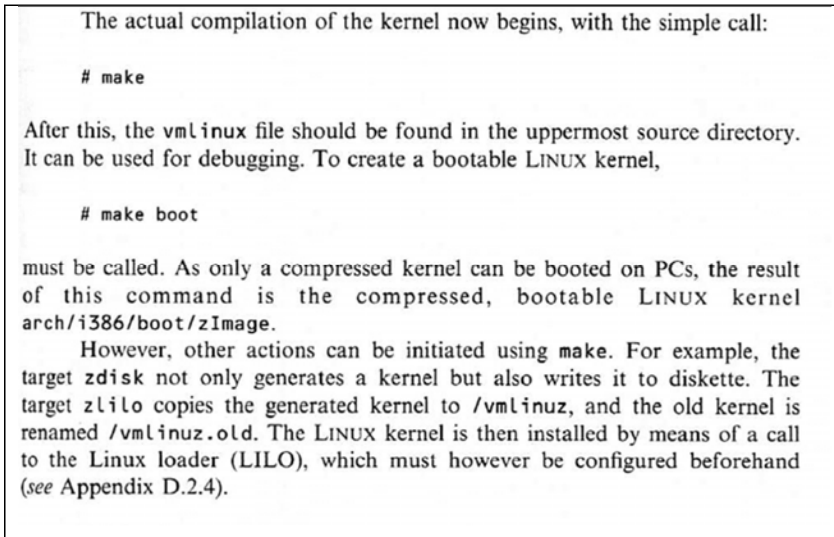


In the text of each section, names of source text commands, variables and so on, are set in a `display` font. Parameters relevant to a special context have been set in an *italic* typeface. For example:

```
% make argument
```

As not all readers of this book will have access to the Internet, slackware distribution 1.2.0 and the German LST distribution 1.7 are supplied on the enclosed CD. Once the appropriate start-up diskettes have been created using the MS-DOS programs **GZIP.EXE** and **RAWRITE.EXE**, these can be installed direct from the CD. The authors would like to express their special thanks to Patrick J. Volkerding and the LINUX support team in Erlangen, in particular Ralf Flaxa and Stefan Probst, for what must have been very time-consuming work on this distribution.

Beck, at Preface



The actual compilation of the kernel now begins, with the simple call:

```
# make
```

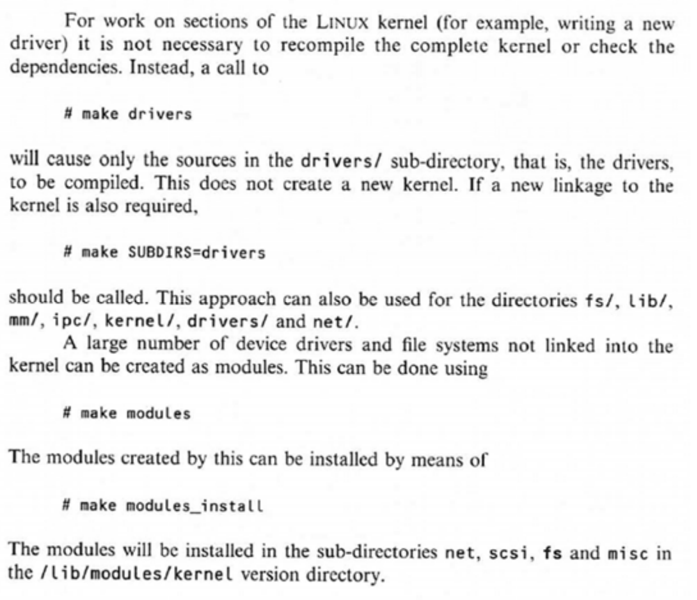
After this, the `vmlinux` file should be found in the uppermost source directory. It can be used for debugging. To create a bootable LINUX kernel,

```
# make boot
```

must be called. As only a compressed kernel can be booted on PCs, the result of this command is the compressed, bootable LINUX kernel `arch/i386/boot/zImage`.

However, other actions can be initiated using `make`. For example, the target `zdisk` not only generates a kernel but also writes it to diskette. The target `zlilo` copies the generated kernel to `/vmlinuz`, and the old kernel is renamed `/vmlinuz.old`. The LINUX kernel is then installed by means of a call to the Linux loader (LILO), which must however be configured beforehand (*see* Appendix D.2.4).

Beck, at 2.2



For work on sections of the LINUX kernel (for example, writing a new driver) it is not necessary to recompile the complete kernel or check the dependencies. Instead, a call to

```
# make drivers
```

will cause only the sources in the `drivers/` sub-directory, that is, the drivers, to be compiled. This does not create a new kernel. If a new linkage to the kernel is also required,

```
# make SUBDIRS=drivers
```

should be called. This approach can also be used for the directories `fs/`, `lib/`, `mm/`, `ipc/`, `kernel/`, `drivers/` and `net/`.

A large number of device drivers and file systems not linked into the kernel can be created as modules. This can be done using

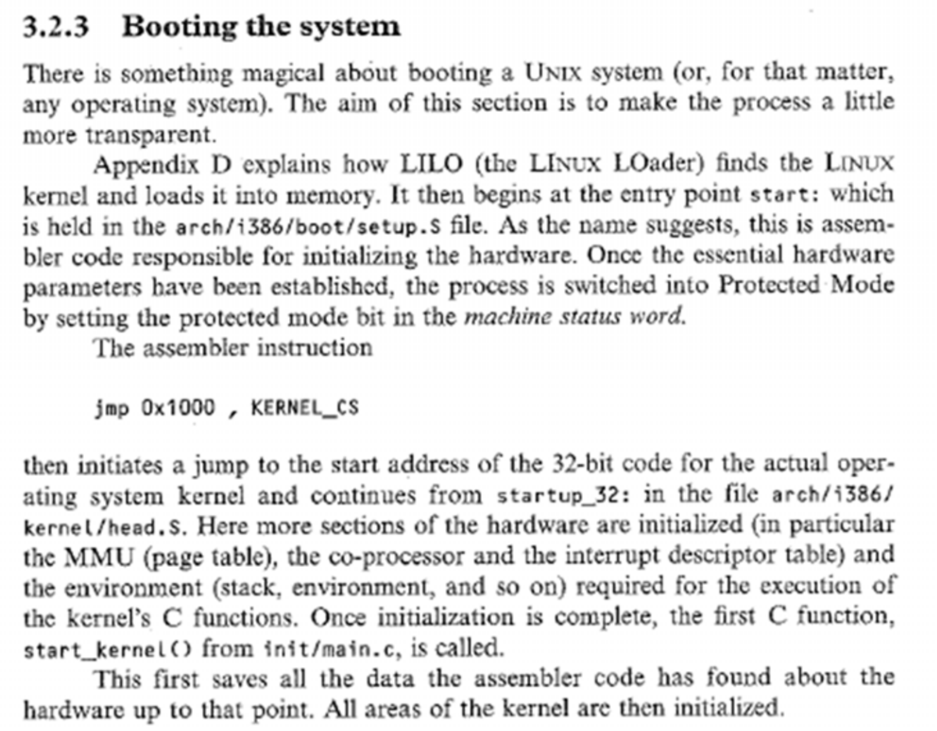
```
# make modules
```

The modules created by this can be installed by means of

```
# make modules_install
```

The modules will be installed in the sub-directories `net`, `scsi`, `fs` and `misc` in the `/lib/modules/kernel` version directory.

Beck, at 2.2



3.2.3 Booting the system

There is something magical about booting a UNIX system (or, for that matter, any operating system). The aim of this section is to make the process a little more transparent.

Appendix D explains how LILO (the LINUX LOader) finds the LINUX kernel and loads it into memory. It then begins at the entry point `start:` which is held in the `arch/i386/boot/setup.S` file. As the name suggests, this is assembler code responsible for initializing the hardware. Once the essential hardware parameters have been established, the process is switched into Protected Mode by setting the protected mode bit in the *machine status word*.

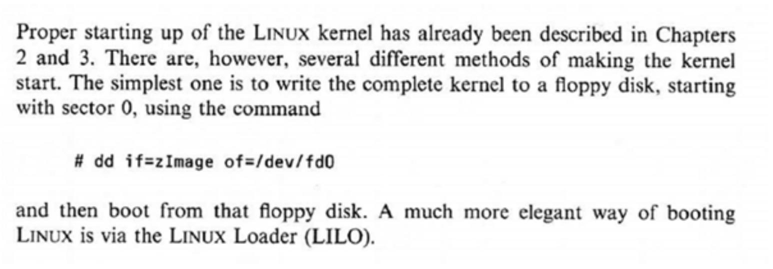
The assembler instruction

```
jmp 0x1000 , KERNEL_CS
```

then initiates a jump to the start address of the 32-bit code for the actual operating system kernel and continues from `startup_32:` in the file `arch/i386/kernel/head.S`. Here more sections of the hardware are initialized (in particular the MMU (page table), the co-processor and the interrupt descriptor table) and the environment (stack, environment, and so on) required for the execution of the kernel's C functions. Once initialization is complete, the first C function, `start_kernel()` from `init/main.c`, is called.

This first saves all the data the assembler code has found about the hardware up to that point. All areas of the kernel are then initialized.

Beck, at 3.2.3

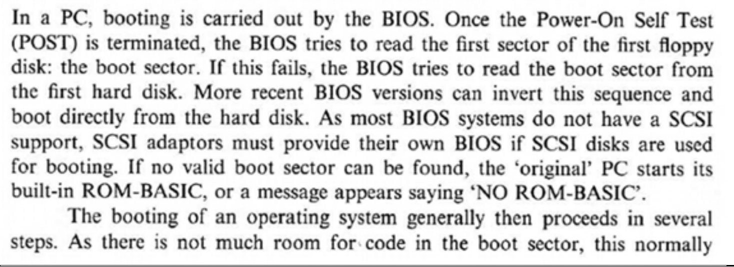


Proper starting up of the LINUX kernel has already been described in Chapters 2 and 3. There are, however, several different methods of making the kernel start. The simplest one is to write the complete kernel to a floppy disk, starting with sector 0, using the command

```
# dd if=zImage of=/dev/fd0
```

and then boot from that floppy disk. A much more elegant way of booting LINUX is via the LINUX Loader (LILO).

Beck, at Appendix D



In a PC, booting is carried out by the BIOS. Once the Power-On Self Test (POST) is terminated, the BIOS tries to read the first sector of the first floppy disk: the boot sector. If this fails, the BIOS tries to read the boot sector from the first hard disk. More recent BIOS versions can invert this sequence and boot directly from the hard disk. As most BIOS systems do not have a SCSI support, SCSI adaptors must provide their own BIOS if SCSI disks are used for booting. If no valid boot sector can be found, the 'original' PC starts its built-in ROM-BASIC, or a message appears saying 'NO ROM-BASIC'.

The booting of an operating system generally then proceeds in several steps. As there is not much room for code in the boot sector, this normally

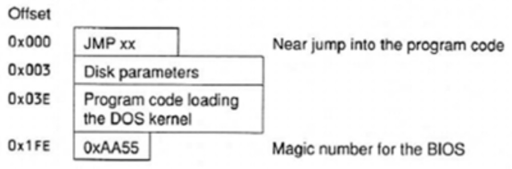

Offset
0x000
JMP xx
Near jump into the program code
0x003
Disk parameters
0x03E
Program code loading the DOS kernel
0x1FE
0xAA55
Magic number for the BIOS

Figure D.1 The MS-DOS boot sector.

loads a second loader, and so on, until the actual operating system kernel is completely loaded.

As Figure D.1 shows, the structure of a boot sector is relatively simple; its length is always 512 bytes (so that it can be stored both on a floppy disk and a hard disk).

In this, the disk parameters are significant only for MS-DOS. It is important that the code starts at offset 0 and that the boot sector is terminated by the *magic number*.

Beck, at Appendix D.1

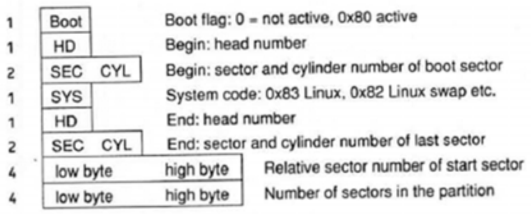

1
Boot
Boot flag: 0 = not active, 0x80 active
1
HD
Begin: head number
2
SEC CYL
Begin: sector and cylinder number of boot sector
1
SYS
System code: 0x83 Linux, 0x82 Linux swap etc.
1
HD
End: head number
2
SEC CYL
End: sector and cylinder number of last sector
4
low byte high byte
Relative sector number of start sector
4
low byte high byte
Number of sectors in the partition

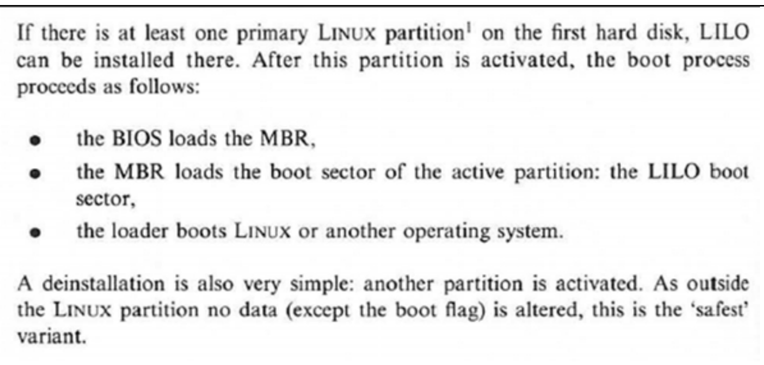
Figure D.3 Structure of a partition entry.

Originally, there were only primary partitions: therefore, `fdisk` under MS-DOS and most of the equivalent programs can only activate those partitions. The code in the MBR must thus only carry out the following operations:

- determine the active partition,
- load the boot sector of the active partition, using the BIOS,
- jump into the boot sector at offset 0.

The number of bytes in the MBR is more than sufficient to do this. Because, as described above, each partition in principle contains a boot sector, and furthermore, the structure of any second hard disk which may be present is similar to that of the first disk, a multitude of replacements for the standard MS-DOS MBR have come about: so-called *boot managers*. They all have in common the fact that they either substitute the MBR with their own code or occupy the boot sector of an active partition. To boot LINUX, most will use the LINUX loader LILO.

Beck, at Appendix D.1

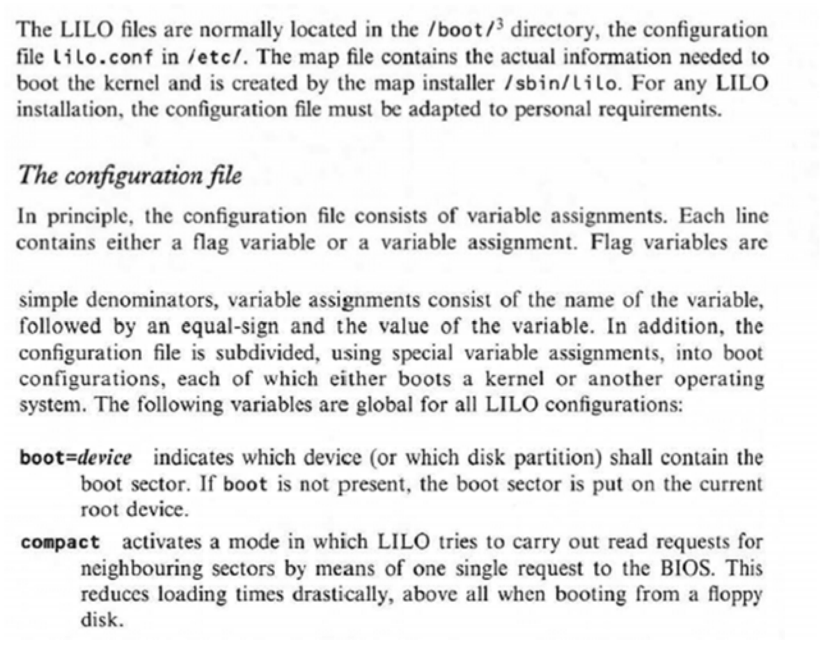


If there is at least one primary LINUX partition[1] on the first hard disk, LILO can be installed there. After this partition is activated, the boot process proceeds as follows:

- the BIOS loads the MBR,
- the MBR loads the boot sector of the active partition: the LILO boot sector,
- the loader boots LINUX or another operating system.

A deinstallation is also very simple: another partition is activated. As outside the LINUX partition no data (except the boot flag) is altered, this is the 'safest' variant.

Beck, at Appendix D.2.1



The LILO files are normally located in the `/boot/`[3] directory, the configuration file `lilo.conf` in `/etc/`. The map file contains the actual information needed to boot the kernel and is created by the map installer `/sbin/lilo`. For any LILO installation, the configuration file must be adapted to personal requirements.

The configuration file

In principle, the configuration file consists of variable assignments. Each line contains either a flag variable or a variable assignment. Flag variables are

simple denominators, variable assignments consist of the name of the variable, followed by an equal-sign and the value of the variable. In addition, the configuration file is subdivided, using special variable assignments, into boot configurations, each of which either boots a kernel or another operating system. The following variables are global for all LILO configurations:

boot=*device* indicates which device (or which disk partition) shall contain the boot sector. If **boot** is not present, the boot sector is put on the current root device.

compact activates a mode in which LILO tries to carry out read requests for neighbouring sectors by means of one single request to the BIOS. This reduces loading times drastically, above all when booting from a floppy disk.

Beck, at Appendix D.2.4

2. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system.	Linux Kernel, as evidenced by the example citations below, discloses "said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an operating system of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Linux Kernel discloses this limitation: *See* Claims 1.1, 1.4, and 1.5 above.	

3. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system.	Linux Kernel, as evidenced by the example citations below, discloses "said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Linux Kernel discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

4. The method of claim 1, wherein said decompressed at least a portion of said boot data comprises program code associated with an application program and an operating system of said computer system.	Linux Kernel, as evidenced by the example citations below, discloses "said decompressed at least a portion of said boot data comprises program code associated with an application program and an operating system of said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said decompressed at least a portion of said boot data comprises program code associated with an application program of said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Linux Kernel discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

5. The method of claim 1, wherein said preloading is performed by a data storage controller connected to said boot device.	Linux Kernel, as evidenced by the example citations below, discloses "said preloading is performed by a data storage controller connected to said boot device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said preloading is performed by a data storage controller connected to said boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Linux Kernel discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

6. The method of claim 1, further comprising updating the list of boot data.	Linux Kernel, as evidenced by the example citations below, discloses "updating the list of boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the list of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Linux Kernel discloses this limitation: *See* Claims 1.1, 1.3, and 1.4 above.	

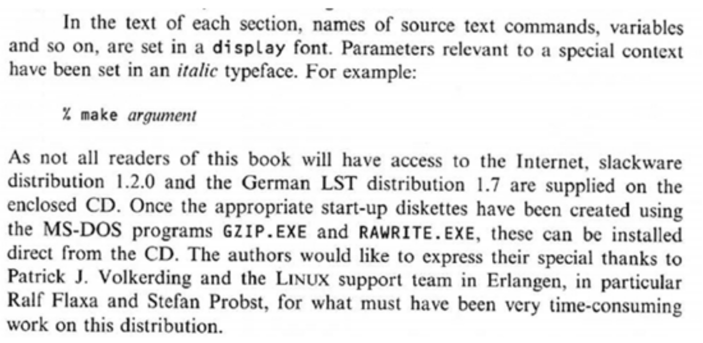
8. The method of claim 1, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form.	Linux Kernel, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in said compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Linux Kernel discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also


In the text of each section, names of source text commands, variables and so on, are set in a `display` font. Parameters relevant to a special context have been set in an *italic* typeface. For example:

```
% make argument
```

As not all readers of this book will have access to the Internet, slackware distribution 1.2.0 and the German LST distribution 1.7 are supplied on the enclosed CD. Once the appropriate start-up diskettes have been created using the MS-DOS programs `GZIP.EXE` and `RAWRITE.EXE`, these can be installed direct from the CD. The authors would like to express their special thanks to Patrick J. Volkerding and the LINUX support team in Erlangen, in particular Ralf Flaxa and Stefan Probst, for what must have been very time-consuming work on this distribution.

Beck, at Preface

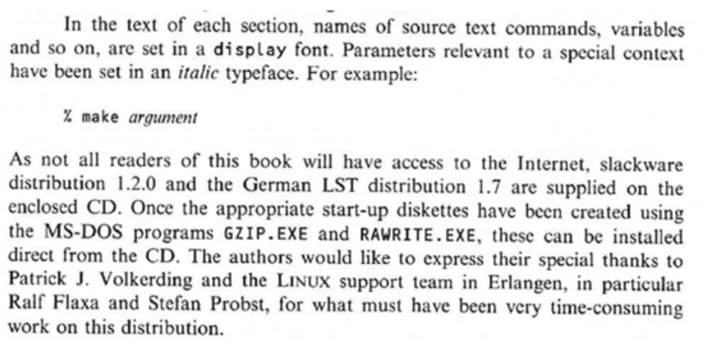
9. The method of claim 1, wherein a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form.	Linux Kernel, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein a plurality of encoders are utilized to provide said at least a portion of compressed data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Linux Kernel discloses this limitation:

See Claims 1.1, 1.3, and 1.4 above.

See also


In the text of each section, names of source text commands, variables and so on, are set in a `display` font. Parameters relevant to a special context have been set in an *italic* typeface. For example:

```
% make argument
```

As not all readers of this book will have access to the Internet, slackware distribution 1.2.0 and the German LST distribution 1.7 are supplied on the enclosed CD. Once the appropriate start-up diskettes have been created using the MS-DOS programs `GZIP.EXE` and `RAWRITE.EXE`, these can be installed direct from the CD. The authors would like to express their special thanks to Patrick J. Volkerding and the LINUX support team in Erlangen, in particular Ralf Flaxa and Stefan Probst, for what must have been very time-consuming work on this distribution.

Beck, at Preface

11.1. a processor;	Linux Kernel, as evidenced by the example citations below, discloses "a processor."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Linux Kernel discloses this limitation: *See* Claim 1.2 above.	

11.2. a memory; and	Linux Kernel, as evidenced by the example citations below, discloses "a memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Linux Kernel discloses this limitation: *See* Claims 1.3, and 1.4 above.	

11.3.1 a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device,	Linux Kernel, as evidenced by the example citations below, discloses "a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a non-volatile memory device for storing logic code associated with the processor, wherein said logic code comprises instructions executable by the processor for maintaining a list of boot data used for booting the host system, at least a portion of said boot data is stored in compressed form in said non-volatile memory device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Linux Kernel discloses this limitation: *See* Claims 1.1, 1.3, 1.4 above.	

11.3.2 said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system; and,	Linux Kernel, as evidenced by the example citations below, discloses "said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said at least a portion of said boot data in compressed form is preloaded into said memory, and said preloaded at least a portion of boot data in compressed form is decompressed and utilized to boot said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Linux Kernel discloses this limitation: *See* Claims 1.3, 1.4, and 1.5 above.	

11.4 a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data.	Linux Kernel, as evidenced by the example citations below, discloses "a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a data compression engine for providing said at least a portion of said boot data in compressed form by compressing said at least a portion of said boot data and decompressing said at least a portion of said boot data in compressed form to provide said decompressed at least a portion of boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Linux Kernel discloses this limitation: *See* Claims 1.1 and 1.5 above.	

12. The system of claim 11, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program.	Linux Kernel, as evidenced by the example citations below, discloses "said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Linux Kernel discloses this limitation: *See* Claims 1.1, 3, and 11.3.1 above.	

Appendix B38
Invalidity of U.S. Patent 8,090,936 based on Linux Kernel

13. The system of claim 11, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system.	Linux Kernel, as evidenced by the example citations below, discloses "wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein said logic code further comprises program instructions executable by said processor for maintaining a list of application data associated with an application program, and wherein said application data is preloaded upon launching the application program and utilized by said computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Linux Kernel discloses this limitation: *See* Claims 1.1, 1.3, 3, 5 and 11.3.1 and 11.3.2 above.	

Appendix B38
Invalidity of U.S. Patent 8,090,936 based on Linux Kernel

15. The system of claim 11, wherein Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form.	Linux Kernel, as evidenced by the example citations below, discloses "Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, Lempel-Ziv encoding is utilized to provide said at least a portion of said boot data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Linux Kernel discloses this limitation: *See* Claims 1, 8, and 11 above.	

16. The system of claim 11, wherein a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form.	Linux Kernel, as evidenced by the example citations below, discloses "a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a plurality of encoders are utilized to provide said at least a portion of said boot data in compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Linux Kernel discloses this limitation: *See* Claims 1, 9, and 11 above.	

Appendix C1
Invalidity of U.S. Patent 8,880,862 based on Esfahani

U.S. Patent Nos. 6,434,695 to Esfahani ("Esfahani '695) and 6,732,265 to Esfahani ("Esfahani '265"), alone or in combination, invalidate claims 1-6, 8-11, 13-17, 19, 23-24, 27-29, 31-33, 35-36, 39-40, 43-45, 47-53, 59-60, 63-65, 67, 71-72, 75-77, 79-81, 83-84, 87-89, 91-93, 97-98, and 107-109 of United States Patent No. 8,880,862 ("the '862 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

In addition, the prior art New World Mac operating system disclosed in Esfahani ("New World Mac System") invalidates claims 1-6, 8-11, 13-17, 19, 23-24, 27-29, 31-33, 35-36, 39-40, 43-45, 47-53, 59-60, 63-65, 67, 71-72, 75-77, 79-81, 83-84, 87-89, 91-93, 97-98, and 107-109 of the '862 Patent pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 because the system was on sale or in public use more than one year prior to the filing date of the '862 patent. Esfahani '695, Esfahani '265, and the New World Mac System are collectively referred to herein as "Esfahani."

Additionally, Apple intends to provide and seek discovery related to this system to obtain copies of relevant materials, including but not limited to, architectural documents, design documents, implementation documents, internal publications, patent applications and business records relating to this product and will supplement these contentions after those materials are discovered or received. Apple also reserves its rights to rely on any additional New World Mac System materials, either individually or collectively, as prior art publications to the '862 patent.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '862 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix C1
Invalidity of U.S. Patent 8,880,862 based on Esfahani

1 (Preamble) A method for providing accelerated loading of an operating system in a computer system, the method comprising:	Esfahani, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, the method comprising:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system in a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

> The low-level portion, including hardware-specific code, is stored in a relatively small read-only memory (ROM), while at least part of the intermediate-level portion is stored as a compressed ROM image on a disk or other mass storage device, which may be located remotely from the computer system. Upon power-up or reset of the computer system, the code in the ROM is executed to read the compressed ROM image into random access memory (RAM) of the computer system. The compressed image is then decompressed and executed as part of the boot sequence. Once decompressed, the portion of RAM storing the intermediate-level code is write-protected in the memory map, and the code in boot ROM is deleted from the memory map. Memory space in RAM that is allocated to the intermediate-level code but not used is returned to the operating system for use as part of system RAM.

Esfahani '695 [Abstract]

> A method and apparatus for use in booting a computer system are provided. The method includes loading a compressed image of a first portion of the OS of the computer system into a storage device of the computer system, which may be RAM, for example. The compressed image of the first portion of the OS is then decompressed and executed as part of the boot sequence of the computer system.

Esfahani '695 2:6-12; *also* Esfahani '265, 2:10-17.

See also Esfahani, Figs. 1, 4, 5, 6A, and 6B.

1.1 loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory;	Esfahani, as evidenced by the example citations below, discloses "loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

> In particular embodiments, the first portion of the operating system may include an intermediate-level portion of the operating system, while a second, low-level portion of the operating system containing hardware specific aspects are stored in read-only memory. The process of transferring the first portion from non-volatile storage to volatile storage and decompressing the first portion may be initiated by the low-level portion stored in the read-only memory. Once decompressed, the portion of memory storing the first portion of the operating system may be write-protected, and this read-only memory portion may be mapped out of the address space of RAM used by the operating system.

Esfahani 695 2:13-25; *also* Esfahani '265, 2:18-29.

> A computer OS using a compressed ROM image in RAM is described. In brief, the low-level portion of an OS of a computer is designed to be separate from the intermediate-level portion of the OS. The low-level portion, which includes hardware-specific code, is stored in a relatively small Boot ROM, while at least part of the intermediate-level portion is stored as a compressed ROM image on a disk or other mass storage device. The mass storage device may be located remotely from the computer system, such as in a file server. Upon power-up or reset of the computer system, the code in the boot ROM is executed to read the compressed ROM image into RAM, i.e., system memory, of the computer system. The compressed image is then decompressed and executed as part of the boot sequence.

Esfahani 695, 2:54-67; *also* Esfahani '265, 2:58-3:4.

> Referring now to FIG. 4, in the improved OS, the low-level (hardware-specific) OS code 31 resides in firmware, in order to handle start-up activities of the computer system. This code fits into one, relatively small ROM, referred to as the Boot ROM 11. Thus, Boot ROM 11 includes all of the hardware-specific code and tables needed to start up the computer as well as to boot the OS and provide common hardware access services the OS might require. Note that the Boot ROM code is not specific to the MacOS or to any other OS. All higher-level software resides elsewhere, as will now be described.
>
> Prior to start-up, the mid-level portion 32 of OS 30 (which corresponds to part of the ToolBox ROM of earlier Macintosh computers) resides in compressed form in a file 40, referred to as Boot Info file 40.

Esfahani ’695, 5:7-23; *also* Esfahani ’265, 5:10-24.

> During start-up, the Boot Info file 40 is loaded into RAM 12, and the compressed mid-level OS 32 is decompressed. Hence, the mid-level OS 32 is essentially a compressed ROM image. The mid-level OS 32 is inserted into the memory map of the computer system as if it were firmware in ROM. That is, the ROM image can be write-protected in the memory map.

Esfahani ’695, 5:44-51; *also* Esfahani ’265, 5:48-54.

> Referring now to FIG. 5, the low-level OS portion 31 and the Boot Info file 40 are illustrated in greater detail. The low-level portion 31 stored in Boot ROM 11 contains the code needed to start up the computer, initialize and examine the hardware, provide a Device Tree (per Open Firmware) to describe the hardware, provide hardware access services, and transfer control to the OS. In one embodiment, the components of the low-level portion 31 include:
>
> > code for performing Power-On Self Test (POST), including code for performing diagnostics, generating a boot beep and an error beep;
> >
> > Open Firmware code;
> >
> > hardware-specific Mac OS drivers (“ndrv's”) that are needed at boot time (drivers needed at boot time, e.g.,video drivers, network drivers, or disk drivers, are loaded from the Device Tree);

> HardwareInit code (i.e., the lowest-level code for initializing the CPU, RAM, clock, system bus, etc.) without Mac OS-specific code;
>
> code for performing Run-Time Abstraction Services (RTAS). Certain hardware devices differ from machine to machine, but provide similar functions. RTAS provides such functions, including functions for accessing the real-time clock, NVRAM 20, restart, shutdown, and PCI configuration cycles. The I/O primitives for these functions in the ROM Image make use of RTAS.

Esfahani ’695, 6:29-54; *also* Esfahani ’265, 6:31-57.

> The Boot Info file 40 will now be described in greater detail. The Boot Info file 40 may be stored in the System Folder of the start-up volume. Alternatively, the Boot Info file 40 may be provided by a network server using, for example, the Bootstrap Protocol (BootP), which is described in B. Croft et al., Network Working Group Request for Comments (RFC) 951, September 1985. Referring to FIG. 5, the Boot Info file 40 includes Open Firmware-specific Mac OS code 52, referred to as the “Trampoline code”; an Open Firmware header 51, which includes a Forth script that performs operations necessary to the start-up of the OS, including validation tests and transfer of control to the Trampoline code; and a compressed ROM Image 53, which represents the mid-level portion 32 of the OS 30. The purpose of the header 51 is, generally, to specify the locations of the other components of the Boot Info file 40. The purpose of the Trampoline code 53 generally is to handle the transition between the Open Firmware code in the Boot Rom 11 and the ROM Image 52, as will be described in greater detail below.

Esfahani ’695, 7:5-24; *also* Esfahani ’265, 7:9:28.

> Note that in alternative embodiments of the OS 30, some or all of the above mentioned components of the ROM image 53 may be provided as separate, compressed elements. These separate elements may be embodied in separate Boot Info files or in a single Boot Info file. This approach would allow the OS components that are required for a given machine to be individually selected, decompressed as part of the ROM image, and used as part of the OS 30, while unnecessary components could be ignored.

Esfahani ’695, 7:43-51; *also* Esfahani ’265, 7:47-55.

See also Esfahani, Figs. 1, 4, 5, 6A, and 6B.

1.2 accessing the loaded portion of the boot data in the compressed form from memory;	Esfahani, as evidenced by the example citations below, discloses "accessing the loaded portion of the boot data in the compressed form from memory;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded portion of the boot data in the compressed form from memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

> In particular embodiments, the first portion of the operating system may include an intermediate-level portion of the operating system, while a second, low-level portion of the operating system containing hardware specific aspects are stored in read-only memory. The process of transferring the first portion from non-volatile storage to volatile storage and decompressing the first portion may be initiated by the low-level portion stored in the read-only memory. Once decompressed, the portion of memory storing the first portion of the operating system may be write-protected, and this read-only memory portion may be mapped out of the address space of RAM used by the operating system.

Esfahani 695 2:13-25; *also* Esfahani '265, 2:18-29.

> A computer OS using a compressed ROM image in RAM is described. In brief, the low-level portion of an OS of a computer is designed to be separate from the intermediate-level portion of the OS. The low-level portion, which includes hardware-specific code, is stored in a relatively small Boot ROM, while at least part of the intermediate-level portion is stored as a compressed ROM image on a disk or other mass storage device. The mass storage device may be located remotely from the computer system, such as in a file server. Upon power-up or reset of the computer system, the code in the boot ROM is executed to read the compressed ROM image into RAM, i.e., system memory, of the computer system. The compressed image is then decompressed and executed as part of the boot sequence.

Esfahani 695, 2:54-67; *also* Esfahani '265, 2:58-3:4.

> Referring now to FIG. 4, in the improved OS, the low-level (hardware-specific) OS code 31 resides in firmware, in order to handle start-up activities of the computer system. This code fits into one, relatively

> small ROM, referred to as the Boot ROM 11. Thus, Boot ROM 11 includes all of the hardware-specific code and tables needed to start up the computer as well as to boot the OS and provide common hardware access services the OS might require. Note that the Boot ROM code is not specific to the MacOS or to any other OS. All higher-level software resides elsewhere, as will now be described.
>
> Prior to start-up, the mid-level portion 32 of OS 30 (which corresponds to part of the ToolBox ROM of earlier Macintosh computers) resides in compressed form in a file 40, referred to as Boot Info file 40.

Esfahani '695, 5:7-23; *also* Esfahani '265, 5:10-24.

> During start-up, the Boot Info file 40 is loaded into RAM 12, and the compressed mid-level OS 32 is decompressed. Hence, the mid-level OS 32 is essentially a compressed ROM image. The mid-level OS 32 is inserted into the memory map of the computer system as if it were firmware in ROM. That is, the ROM image can be write-protected in the memory map.

Esfahani '695, 5:44-51; *also* Esfahani '265, 5:48-54.

> Referring now to FIG. 5, the low-level OS portion 31 and the Boot Info file 40 are illustrated in greater detail. The low-level portion 31 stored in Boot ROM 11 contains the code needed to start up the computer, initialize and examine the hardware, provide a Device Tree (per Open Firmware) to describe the hardware, provide hardware access services, and transfer control to the OS. In one embodiment, the components of the low-level portion 31 include:
>
> > code for performing Power-On Self Test (POST), including code for performing diagnostics, generating a boot beep and an error beep;
> >
> > Open Firmware code;
> >
> > hardware-specific Mac OS drivers ("ndrv's") that are needed at boot time (drivers needed at boot time, e.g.,video drivers, network drivers, or disk drivers, are loaded from the Device Tree);
> >
> > HardwareInit code (i.e., the lowest-level code for initializing the CPU, RAM, clock, system bus, etc.) without Mac OS-specific code;

> code for performing Run-Time Abstraction Services (RTAS). Certain hardware devices differ from machine to machine, but provide similar functions. RTAS provides such functions, including functions for accessing the real-time clock, NVRAM 20, restart, shutdown, and PCI configuration cycles. The I/O primitives for these functions in the ROM Image make use of RTAS.

Esfahani ’695, 6:29-54; *also* Esfahani ’265, 6:31-57.

> The Boot Info file 40 will now be described in greater detail. The Boot Info file 40 may be stored in the System Folder of the start-up volume. Alternatively, the Boot Info file 40 may be provided by a network server using, for example, the Bootstrap Protocol (BootP), which is described in B. Croft et al., Network Working Group Request for Comments (RFC) 951, September 1985. Referring to FIG. 5, the Boot Info file 40 includes Open Firmware-specific Mac OS code 52, referred to as the “Trampoline code”; an Open Firmware header 51, which includes a Forth script that performs operations necessary to the start-up of the OS, including validation tests and transfer of control to the Trampoline code; and a compressed ROM Image 53, which represents the mid-level portion 32 of the OS 30. The purpose of the header 51 is, generally, to specify the locations of the other components of the Boot Info file 40. The purpose of the Trampoline code 53 generally is to handle the transition between the Open Firmware code in the Boot Rom 11 and the ROM Image 52, as will be described in greater detail below.

Esfahani ’695, 7:5-24; *also* Esfahani ’265, 7:9:28.

> Note that in alternative embodiments of the OS 30, some or all of the above mentioned components of the ROM image 53 may be provided as separate, compressed elements. These separate elements may be embodied in separate Boot Info files or in a single Boot Info file. This approach would allow the OS components that are required for a given machine to be individually selected, decompressed as part of the ROM image, and used as part of the OS 30, while unnecessary components could be ignored.

Esfahani ’695, 7:43-51; *also* Esfahani ’265, 7:47-55.

See also Esfahani, Figs. 1, 4, 5, 6A, and 6B.

1.3 decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form; and	Esfahani, as evidenced by the example citations below, discloses "decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

> In particular embodiments, the first portion of the operating system may include an intermediate-level portion of the operating system, while a second, low-level portion of the operating system containing hardware specific aspects are stored in read-only memory. The process of transferring the first portion from non-volatile storage to volatile storage and decompressing the first portion may be initiated by the low-level portion stored in the read-only memory. Once decompressed, the portion of memory storing the first portion of the operating system may be write-protected, and this read-only memory portion may be mapped out of the address space of RAM used by the operating system.

Esfahani 695 2:13-25; *also* Esfahani '265, 2:18-29.

> A computer OS using a compressed ROM image in RAM is described. In brief, the low-level portion of an OS of a computer is designed to be separate from the intermediate-level portion of the OS. The low-level portion, which includes hardware-specific code, is stored in a relatively small Boot ROM, while at least part of the intermediate-level portion is stored as a compressed ROM image on a disk or other mass storage device. The mass storage device may be located remotely from the computer system, such as in a file server. Upon power-up or reset of the computer system, the code in the boot ROM is executed to read the compressed ROM image into RAM, i.e., system memory, of the

> computer system. The compressed image is then decompressed and executed as part of the boot sequence.

Esfahani 695, 2:54-67; *also* Esfahani '265, 2:58-3:4.

> Referring now to FIG. 4, in the improved OS, the low-level (hardware-specific) OS code 31 resides in firmware, in order to handle start-up activities of the computer system. This code fits into one, relatively small ROM, referred to as the Boot ROM 11. Thus, Boot ROM 11 includes all of the hardware-specific code and tables needed to start up the computer as well as to boot the OS and provide common hardware access services the OS might require. Note that the Boot ROM code is not specific to the MacOS or to any other OS. All higher-level software resides elsewhere, as will now be described.
>
> Prior to start-up, the mid-level portion 32 of OS 30 (which corresponds to part of the ToolBox ROM of earlier Macintosh computers) resides in compressed form in a file 40, referred to as Boot Info file 40.

Esfahani '695, 5:7-23; *also* Esfahani '265, 5:10-24.

> During start-up, the Boot Info file 40 is loaded into RAM 12, and the compressed mid-level OS 32 is decompressed. Hence, the mid-level OS 32 is essentially a compressed ROM image. The mid-level OS 32 is inserted into the memory map of the computer system as if it were firmware in ROM. That is, the ROM image can be write-protected in the memory map.

Esfahani '695, 5:44-51; *also* Esfahani '265, 5:48-54.

> Referring now to FIG. 5, the low-level OS portion 31 and the Boot Info file 40 are illustrated in greater detail. The low-level portion 31 stored in Boot ROM 11 contains the code needed to start up the computer, initialize and examine the hardware, provide a Device Tree (per Open Firmware) to describe the hardware, provide hardware access services, and transfer control to the OS. In one embodiment, the components of the low-level portion 31 include:
>
>> code for performing Power-On Self Test (POST), including code for performing diagnostics, generating a boot beep and an error beep;
>>
>> Open Firmware code;

> hardware-specific Mac OS drivers ("ndrv's") that are needed at boot time (drivers needed at boot time, e.g.,video drivers, network drivers, or disk drivers, are loaded from the Device Tree);
>
> HardwareInit code (i.e., the lowest-level code for initializing the CPU, RAM, clock, system bus, etc.) without Mac OS-specific code;
>
> code for performing Run-Time Abstraction Services (RTAS). Certain hardware devices differ from machine to machine, but provide similar functions. RTAS provides such functions, including functions for accessing the real-time clock, NVRAM 20, restart, shutdown, and PCI configuration cycles. The I/O primitives for these functions in the ROM Image make use of RTAS.

Esfahani '695, 6:29-54; *also* Esfahani '265, 6:31-57.

> The Boot Info file 40 will now be described in greater detail. The Boot Info file 40 may be stored in the System Folder of the start-up volume. Alternatively, the Boot Info file 40 may be provided by a network server using, for example, the Bootstrap Protocol (BootP), which is described in B. Croft et al., Network Working Group Request for Comments (RFC) 951, September 1985. Referring to FIG. 5, the Boot Info file 40 includes Open Firmware-specific Mac OS code 52, referred to as the "Trampoline code"; an Open Firmware header 51, which includes a Forth script that performs operations necessary to the start-up of the OS, including validation tests and transfer of control to the Trampoline code; and a compressed ROM Image 53, which represents the mid-level portion 32 of the OS 30. The purpose of the header 51 is, generally, to specify the locations of the other components of the Boot Info file 40. The purpose of the Trampoline code 53 generally is to handle the transition between the Open Firmware code in the Boot Rom 11 and the ROM Image 52, as will be described in greater detail below.

Esfahani '695, 7:5-24; *also* Esfahani '265, 7:9:28.

> Note that in alternative embodiments of the OS 30, some or all of the above mentioned components of the ROM image 53 may be provided as separate, compressed elements. These separate elements may be embodied in separate Boot Info files or in a single Boot Info file. This approach would allow the OS components that are required for a given machine to be individually selected, decompressed as part of the ROM

> image, and used as part of the OS 30, while unnecessary components could be ignored.

Esfahani ’695, 7:43-51; *also* Esfahani ’265, 7:47-55.

See also Esfahani, Figs. 1, 4, 5, 6A, and 6B..

1.4.1 updating the boot data list,	Esfahani, as evidenced by the example citations below, discloses "updating the boot data list,"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

> Referring now to FIG. 4, in the improved OS, the low-level (hardware-specific) OS code 31 resides in firmware, in order to handle start-up activities of the computer system. This code fits into one, relatively small ROM, referred to as the Boot ROM 11. Thus, Boot ROM 11 includes all of the hardware-specific code and tables needed to start up the computer as well as to boot the OS and provide common hardware access services the OS might require. Note that the Boot ROM code is not specific to the MacOS or to any other OS. All higher-level software resides elsewhere, as will now be described.
>
> Prior to start-up, the mid-level portion 32 of OS 30 (which corresponds to part of the ToolBox ROM of earlier Macintosh computers) resides in compressed form in a file 40, referred to as Boot Info file 40.

Esfahani '695, 5:7-23; *also* Esfahani '265, 5:10-24.

> During start-up, the Boot Info file 40 is loaded into RAM 12, and the compressed mid-level OS 32 is decompressed. Hence, the mid-level OS 32 is essentially a compressed ROM image. The mid-level OS 32 is inserted into the memory map of the computer system as if it were firmware in ROM. That is, the ROM image can be write-protected in the memory map.

Esfahani '695, 5:44-51; *also* Esfahani '265, 5:48-54.

> Referring now to FIG. 5, the low-level OS portion 31 and the Boot Info file 40 are illustrated in greater detail. The low-level portion 31 stored in Boot ROM 11 contains the code needed to start up the computer, initialize and examine the hardware, provide a Device Tree (per Open Firmware) to describe the hardware, provide hardware access services, and transfer control to the OS. In one embodiment, the components of the low-level portion 31 include:

> code for performing Power-On Self Test (POST), including code for performing diagnostics, generating a boot beep and an error beep;
>
> Open Firmware code;
>
> hardware-specific Mac OS drivers (“ndrv's”) that are needed at boot time (drivers needed at boot time, e.g.,video drivers, network drivers, or disk drivers, are loaded from the Device Tree);
>
> HardwareInit code (i.e., the lowest-level code for initializing the CPU, RAM, clock, system bus, etc.) without Mac OS-specific code;
>
> code for performing Run-Time Abstraction Services (RTAS). Certain hardware devices differ from machine to machine, but provide similar functions. RTAS provides such functions, including functions for accessing the real-time clock, NVRAM 20, restart, shutdown, and PCI configuration cycles. The I/O primitives for these functions in the ROM Image make use of RTAS.

Esfahani ’695, 6:29-54; *also* Esfahani ’265, 6:31-57.

> The Boot Info file 40 will now be described in greater detail. The Boot Info file 40 may be stored in the System Folder of the start-up volume. Alternatively, the Boot Info file 40 may be provided by a network server using, for example, the Bootstrap Protocol (BootP), which is described in B. Croft et al., Network Working Group Request for Comments (RFC) 951, September 1985. Referring to FIG. 5, the Boot Info file 40 includes Open Firmware-specific Mac OS code 52, referred to as the “Trampoline code”; an Open Firmware header 51, which includes a Forth script that performs operations necessary to the start-up of the OS, including validation tests and transfer of control to the Trampoline code; and a compressed ROM Image 53, which represents the mid-level portion 32 of the OS 30. The purpose of the header 51 is, generally, to specify the locations of the other components of the Boot Info file 40. The purpose of the Trampoline code 53 generally is to handle the transition between the Open Firmware code in the Boot Rom 11 and the ROM Image 52, as will be described in greater detail below.

Esfahani ’695, 7:5-24; *also* Esfahani ’265, 7:9:28.

> Note that in alternative embodiments of the OS 30, some or all of the above mentioned components of the ROM image 53 may be provided as

> separate, compressed elements. These separate elements may be embodied in separate Boot Info files or in a single Boot Info file. This approach would allow the OS components that are required for a given machine to be individually selected, decompressed as part of the ROM image, and used as part of the OS 30, while unnecessary components could be ignored.

Esfahani ’695, 7:43-51; *also* Esfahani ’265, 7:47-55.

> The Boot Info file 40 resides on the boot device (e.g., a disk, or on a network) and has a localizable name.

Esfahani ’695, 7:66-67; *also* Esfahani ’265, 8:3-4.

See also Esfahani, Figs. 1, 4, 5, 6A, and 6B.

1.4.2 wherein the decompressed portion of boot data comprises a portion of the operating system.	Esfahani, as evidenced by the example citations below, discloses "wherein the decompressed portion of boot data comprises a portion of the operating system."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the decompressed portion of boot data comprises a portion of the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

> In particular embodiments, the first portion of the operating system may include an intermediate-level portion of the operating system, while a second, low-level portion of the operating system containing hardware specific aspects are stored in read-only memory. The process of transferring the first portion from non-volatile storage to volatile storage and decompressing the first portion may be initiated by the low-level portion stored in the read-only memory. Once decompressed, the portion of memory storing the first portion of the operating system may be write-protected, and this read-only memory portion may be mapped out of the address space of RAM used by the operating system.

Esfahani 695 2:13-25; *also* Esfahani '265, 2:18-29.

> A computer OS using a compressed ROM image in RAM is described. In brief, the low-level portion of an OS of a computer is designed to be separate from the intermediate-level portion of the OS. The low-level portion, which includes hardware-specific code, is stored in a relatively small Boot ROM, while at least part of the intermediate-level portion is stored as a compressed ROM image on a disk or other mass storage device. The mass storage device may be located remotely from the computer system, such as in a file server. Upon power-up or reset of the computer system, the code in the boot ROM is executed to read the compressed ROM image into RAM, i.e., system memory, of the computer system. The compressed image is then decompressed and executed as part of the boot sequence.

Esfahani 695, 2:54-67; *also* Esfahani '265, 2:58-3:4.

> Referring now to FIG. 4, in the improved OS, the low-level (hardware-specific) OS code 31 resides in firmware, in order to handle start-up activities of the computer system. This code fits into one, relatively

> small ROM, referred to as the Boot ROM 11. Thus, Boot ROM 11 includes all of the hardware-specific code and tables needed to start up the computer as well as to boot the OS and provide common hardware access services the OS might require. Note that the Boot ROM code is not specific to the MacOS or to any other OS. All higher-level software resides elsewhere, as will now be described.
>
> Prior to start-up, the mid-level portion 32 of OS 30 (which corresponds to part of the ToolBox ROM of earlier Macintosh computers) resides in compressed form in a file 40, referred to as Boot Info file 40.

Esfahani ’695, 5:7-23; *also* Esfahani ’265, 5:10-24.

> During start-up, the Boot Info file 40 is loaded into RAM 12, and the compressed mid-level OS 32 is decompressed. Hence, the mid-level OS 32 is essentially a compressed ROM image. The mid-level OS 32 is inserted into the memory map of the computer system as if it were firmware in ROM. That is, the ROM image can be write-protected in the memory map.

Esfahani ’695, 5:44-51; *also* Esfahani ’265, 5:48-54.

See also Esfahani, Figs. 1, 4, 5, 6A, and 6B.

2. The method of claim 1, wherein the updating comprises: associating additional boot data with the boot data list.	Esfahani, as evidenced by the example citations below, discloses "wherein the updating comprises: associating additional boot data with the boot data list."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein associating additional boot data with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

See Claim 1.4.1 above.

See also

> In particular embodiments, the first portion of the operating system may include an intermediate-level portion of the operating system, while a second, low-level portion of the operating system containing hardware specific aspects are stored in read-only memory. The process of transferring the first portion from non-volatile storage to volatile storage and decompressing the first portion may be initiated by the low-level portion stored in the read-only memory. Once decompressed, the portion of memory storing the first portion of the operating system may be write-protected, and this read-only memory portion may be mapped out of the address space of RAM used by the operating system.

Esfahani, '695, 2:14-25; Esfahani '265, 2:17-29

> Note that in alternative embodiments of the OS 30, some or all of the above mentioned components of the ROM image **53** may be provided as separate, compressed elements. These separate elements may be embodied in separate Boot Info files or in a single Boot Info file. This approach would allow the OS components that are required for a given machine to be individually selected, decompressed as part of the ROM image, and used as part of the OS 30, while unnecessary components could be ignored. For example, components which might be embodied separately include the kernel, the 68K Emulator, particular drivers or other software components that are not common to all Macintosh computers, and code libraries. Hence, this approach may reduce the amount of RAM consumed by the OS. For example, if components are stored in separate boot Info files, then only the necessary Boot Info files would need to be read into RAM during the boot process. Alternatively,

> if components are embodied as discrete components within a single Boot Info file, then the system might read the Boot Info file into RAM, identify the needed components, decompress only the needed components into a separate memory space, and return the unused memory space to the OS for use as system RAM.

Esfahani, ’695, 7:43-65; Esfahani ’265, 7:47-8:2

3. The method of claim 1, wherein the updating comprises: removing an association of additional boot data that is associated with the boot data list from the boot data list.	Esfahani, as evidenced by the example citations below, discloses "wherein the updating comprises: removing an association of additional boot data that is associated with the boot data list from the boot data list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein removing an association of additional boot data that is associated with the boot data list from the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1.4.1 above.	

4. The method of claim 1, wherein the updating comprises: associating additional boot data with the boot data list; and compressing a portion of the additional boot data.	Esfahani, as evidenced by the example citations below, discloses "wherein the updating comprises: associating additional boot data with the boot data list; and compressing a portion of the additional boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein associating additional boot data with the boot data list; and compressing a portion of the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

See Claims 1.4.1, and 2 above.

See also

> In particular embodiments, the first portion of the operating system may include an intermediate-level portion of the operating system, while a second, low-level portion of the operating system containing hardware specific aspects are stored in read-only memory. The process of transferring the first portion from non-volatile storage to volatile storage and decompressing the first portion may be initiated by the low-level portion stored in the read-only memory. Once decompressed, the portion of memory storing the first portion of the operating system may be write-protected, and this read-only memory portion may be mapped out of the address space of RAM used by the operating system.

Esfahani, '695, 2:14-25; Esfahani '265, 2:17-29

> Note that in alternative embodiments of the OS 30, some or all of the above mentioned components of the ROM image **53** may be provided as separate, compressed elements. These separate elements may be embodied in separate Boot Info files or in a single Boot Info file. This approach would allow the OS components that are required for a given machine to be individually selected, decompressed as part of the ROM image, and used as part of the OS 30, while unnecessary components could be ignored. For example, components which might be embodied separately include the kernel, the 68K Emulator, particular drivers or other software components that are not common to all Macintosh computers, and code libraries. Hence, this approach may reduce the amount of RAM consumed by the OS. For example, if components are stored in separate boot Info files, then only the necessary Boot Info files

would need to be read into RAM during the boot process. Alternatively, if components are embodied as discrete components within a single Boot Info file, then the system might read the Boot Info file into RAM, identify the needed components, decompress only the needed components into a separate memory space, and return the unused memory space to the OS for use as system RAM. Esfahani, '695, 7:43-65; Esfahani '265, 7:47-8:2

5 (Preamble) A method for booting a computer system, the method comprising:	Esfahani, as evidenced by the example citations below, discloses "a method for booting a computer system, the method comprising:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

See Claims 1-1.4.2 above.

See also

> Referring now to FIG. 5, the low-level OS portion 31 and the Boot Info file 40 are illustrated in greater detail. The low-level portion 31 stored in Boot ROM 11 contains the code needed to start up the computer, initialize and examine the hardware, provide a Device Tree (per Open Firmware) to describe the hardware, provide hardware access services, and transfer control to the OS. In one embodiment, the components of the low-level portion 31 include:
>
>> code for performing Power-On Self Test (POST), including code for performing diagnostics, generating a boot beep and an error beep;
>>
>> Open Firmware code;
>>
>> hardware-specific Mac OS drivers ("ndrv's") that are needed at boot time (drivers needed at boot time, e.g.,video drivers, network drivers, or disk drivers, are loaded from the Device Tree);
>>
>> HardwareInit code (i.e., the lowest-level code for initializing the CPU, RAM, clock, system bus, etc.) without Mac OS-specific code;
>>
>> code for performing Run-Time Abstraction Services (RTAS). Certain hardware devices differ from machine to machine, but provide similar functions. RTAS provides such functions, including functions for accessing the real-time clock, NVRAM 20, restart, shutdown, and PCI configuration cycles. The I/O

> primitives for these functions in the ROM Image make use of RTAS.

Esfahani ’695, 6:29-54; *also* Esfahani ’265, 6:31-57.

> Overall Boot Process
>
> Referring now to FIGS. 5, 6A and 6B, the overall boot process of the improved OS 30 will now be described. In response to power on or reset at block 601 (FIG. 6A), the POST code executes (preliminary diagnostics, boot beep, initialization) at 602. At 603, the Open Firmware routine initializes and begins execution, including building the expanded Device Tree and the Interrupt Trees. At 604, the Open Firmware locates the Boot Info file 40, based on defaults and NVRAM settings, and loads the Boot Info file 40 into RAM 12. At 605, the Open Firmware executes the Forth script in the Boot Info file 40, which contains information about the rest of the file (offsets, etc.) and instructions to read both the Trampoline code 52 and the compressed ROM Image 53, and places them into a temporary place in RAM 12. At 606, the Open Firmware transfers control to the Trampoline code 52. At 607, the Trampoline code 52 decompresses the ROM Image 53 in RAM 12. In addition, the Trampoline code reallocates any unused memory space in the ROM Image 52; gathers information about the system from Open Firmware; creates data structures based on this information; terminates Open Firmware, and rearranges the contents of memory to an interim location in physical memory space. At 608, the Trampoline code 52 transfers control to the HardwareInit routine of the (now decompressed) ROM Image 53. At 609, the HardwareInit routine copies data structures to their correct places in memory and then calls the kernel of the OS. Next, at 610 the kernel fills in its data structures and then calls the 68K Emulator. At 611, the 68K Emulator initializes itself and then transfers control to the StartInit routine. At 612, the StartInit routine begins execution, initializing data structures and managers, and booting the MacOS (the high-level portion of the OS). Note that blocks 609 through 612, above, are specific to a Mac OS.

Esfahani ’695, 9:38-10:6; *also* Esfahani ’265, 8:42-9:10.

See also Esfahani, Figs. 1, 4, 5, 6A, and 6B.

5.1 storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory;	Esfahani, as evidenced by the example citations below, discloses “storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1.1 above.	

5.2 loading the stored compressed boot data from the first memory;	Esfahani, as evidenced by the example citations below, discloses "loading the stored compressed boot data from the first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the stored compressed boot data from the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1.1 above.	

5.3 accessing the loaded compressed boot data;	Esfahani, as evidenced by the example citations below, discloses "accessing the loaded compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1.2 above.	

5.4 decompressing the accessed compressed boot data;	Esfahani, as evidenced by the example citations below, discloses "decompressing the accessed compressed boot data"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

See Claim 1.3 above.

5.5 utilizing the decompressed boot data to at least partially boot the computer system;	Esfahani, as evidenced by the example citations below, discloses "utilizing the decompressed boot data to at least partially boot the computer system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed boot data to at least partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 1.3 and 1.4.2 above. *See also* > The low-level portion, including hardware-specific code, is stored in a relatively small read-only memory (ROM), while at least part of the intermediate-level portion is stored as a compressed ROM image on a disk or other mass storage device, which may be located remotely from the computer system. Upon power-up or reset of the computer system, the code in the ROM is executed to read the compressed ROM image into random access memory (RAM) of the computer system. The compressed image is then decompressed and executed as part of the boot sequence. Once decompressed, the portion of RAM storing the intermediate-level code is write-protected in the memory map, and the code in boot ROM is deleted from the memory map. Memory space in RAM that is allocated to the intermediate-level code but not used is returned to the operating system for use as part of system RAM. Esfahani '695 [Abstract] > A method and apparatus for use in booting a computer system are provided. The method includes loading a compressed image of a first portion of the OS of the computer system into a storage device of the computer system, which may be RAM, for example. The compressed image of the first portion of the OS is then decompressed and executed as part of the boot sequence of the computer system. Esfahani '695 2:6-12; *also* Esfahani '265, 2:10-17. > In particular embodiments, the first portion of the operating system may include an intermediate-level portion of the operating system, while a second, low-level portion of the operating system containing hardware	

specific aspects are stored in read-only memory. The process of transferring the first portion from non-volatile storage to volatile storage and decompressing the first portion may be initiated by the low-level portion stored in the read-only memory. Once decompressed, the portion of memory storing the first portion of the operating system may be write-protected, and this read-only memory portion may be mapped out of the address space of RAM used by the operating system.

Esfahani 695 2:13-25; *also* Esfahani '265, 2:18-29.

A computer OS using a compressed ROM image in RAM is described. In brief, the low-level portion of an OS of a computer is designed to be separate from the intermediate-level portion of the OS. The low-level portion, which includes hardware-specific code, is stored in a relatively small Boot ROM, while at least part of the intermediate-level portion is stored as a compressed ROM image on a disk or other mass storage device. The mass storage device may be located remotely from the computer system, such as in a file server. Upon power-up or reset of the computer system, the code in the boot ROM is executed to read the compressed ROM image into RAM, i.e., system memory, of the computer system. The compressed image is then decompressed and executed as part of the boot sequence.

Esfahani 695, 2:54-67; *also* Esfahani '265, 2:58-3:4.

Note that in alternative embodiments of the OS 30, some or all of the above mentioned components of the ROM image 53 may be provided as separate, compressed elements. These separate elements may be embodied in separate Boot Info files or in a single Boot Info file. This approach would allow the OS components that are required for a given machine to be individually selected, decompressed as part of the ROM image, and used as part of the OS 30, while unnecessary components could be ignored.

Esfahani '695, 7:43-51; *also* Esfahani '265, 7:47-55.

Overall Boot Process

Referring now to FIGS. 5, 6A and 6B, the overall boot process of the improved OS 30 will now be described. In response to power on or reset at block 601 (FIG. 6A), the POST code executes (preliminary diagnostics, boot beep, initialization) at 602. At 603, the Open Firmware routine initializes and begins execution, including building the expanded Device Tree and the Interrupt Trees. At 604, the Open Firmware locates the Boot Info file 40, based on defaults and NVRAM settings, and loads the Boot Info file 40 into RAM 12. At 605, the Open

> Firmware executes the Forth script in the Boot Info file 40, which contains information about the rest of the file (offsets, etc.) and instructions to read both the Trampoline code 52 and the compressed ROM Image 53, and places them into a temporary place in RAM 12. At 606, the Open Firmware transfers control to the Trampoline code 52. At 607, the Trampoline code 52 decompresses the ROM Image 53 in RAM 12. In addition, the Trampoline code reallocates any unused memory space in the ROM Image 52; gathers information about the system from Open Firmware; creates data structures based on this information; terminates Open Firmware, and rearranges the contents of memory to an interim location in physical memory space. At 608, the Trampoline code 52 transfers control to the HardwareInit routine of the (now decompressed) ROM Image 53. At 609, the HardwareInit routine copies data structures to their correct places in memory and then calls the kernel of the OS. Next, at 610 the kernel fills in its data structures and then calls the 68K Emulator. At 611, the 68K Emulator initializes itself and then transfers control to the StartInit routine. At 612, the StartInit routine begins execution, initializing data structures and managers, and booting the MacOS (the high-level portion of the OS). Note that blocks 609 through 612, above, are specific to a Mac OS.

Esfahani ’695, 9:38-10:6; *also* Esfahani ’265, 8:42-9:10.

See also Esfahani, Figs. 1, 4, 5, 6A, and 6B.

5.6 updating the boot data list;	Esfahani, as evidenced by the example citations below, discloses “updating the boot data list”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1.4.1 above.	

5.7 wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form.	Esfahani, as evidenced by the example citations below, discloses "wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 1-1.3 above.	

6 (Preamble) A system comprising: a processor;	Esfahani, as evidenced by the example citations below, discloses "a system comprising: a processor:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system comprising: a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

> Referring now to FIG. 5, the low-level OS portion 31 and the Boot Info file 40 are illustrated in greater detail. The low-level portion 31 stored in Boot ROM 11 contains the code needed to start up the computer, initialize and examine the hardware, provide a Device Tree (per Open Firmware) to describe the hardware, provide hardware access services, and transfer control to the OS. In one embodiment, the components of the low-level portion 31 include:
>
> > code for performing Power-On Self Test (POST), including code for performing diagnostics, generating a boot beep and an error beep;
> >
> > Open Firmware code;
> >
> > hardware-specific Mac OS drivers ("ndrv's") that are needed at boot time (drivers needed at boot time, e.g.,video drivers, network drivers, or disk drivers, are loaded from the Device Tree);
> >
> > HardwareInit code (i.e., the lowest-level code for initializing the CPU, RAM, clock, system bus, etc.) without Mac OS-specific code;
> >
> > code for performing Run-Time Abstraction Services (RTAS). Certain hardware devices differ from machine to machine, but provide similar functions. RTAS provides such functions, including functions for accessing the real-time clock, NVRAM 20, restart, shutdown, and PCI configuration cycles. The I/O primitives for these functions in the ROM Image make use of RTAS.

Esfahani '695, 6:29-54; *also* Esfahani '265, 6:31-57.

> Overall Boot Process
>
> Referring now to FIGS. 5, 6A and 6B, the overall boot process of the improved OS 30 will now be described. In response to power on or reset at block 601 (FIG. 6A), the POST code executes (preliminary diagnostics, boot beep, initialization) at 602. At 603, the Open Firmware routine initializes and begins execution, including building the expanded Device Tree and the Interrupt Trees. At 604, the Open Firmware locates the Boot Info file 40, based on defaults and NVRAM settings, and loads the Boot Info file 40 into RAM 12. At 605, the Open Firmware executes the Forth script in the Boot Info file 40, which contains information about the rest of the file (offsets, etc.) and instructions to read both the Trampoline code 52 and the compressed ROM Image 53, and places them into a temporary place in RAM 12. At 606, the Open Firmware transfers control to the Trampoline code 52. At 607, the Trampoline code 52 decompresses the ROM Image 53 in RAM 12. In addition, the Trampoline code reallocates any unused memory space in the ROM Image 52; gathers information about the system from Open Firmware; creates data structures based on this information; terminates Open Firmware, and rearranges the contents of memory to an interim location in physical memory space. At 608, the Trampoline code 52 transfers control to the HardwareInit routine of the (now decompressed) ROM Image 53. At 609, the HardwareInit routine copies data structures to their correct places in memory and then calls the kernel of the OS. Next, at 610 the kernel fills in its data structures and then calls the 68K Emulator. At 611, the 68K Emulator initializes itself and then transfers control to the StartInit routine. At 612, the StartInit routine begins execution, initializing data structures and managers, and booting the MacOS (the high-level portion of the OS). Note that blocks 609 through 612, above, are specific to a Mac OS.

Esfahani ’695, 9:38-10:6; *also* Esfahani ’265, 8:42-9:10.

See also Esfahani, Figs. 1, 4, 5, 6A, and 6B.

6.1 a memory; and	Esfahani, as evidenced by the example citations below, discloses "a memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a memory.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 1.1 and 1.2 above.	

6.2 a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor	Esfahani, as evidenced by the example citations below, discloses "a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

See Claims 1.1, 1.2, and Claim 6 (Preamble) above.

See also

> In particular embodiments, the first portion of the operating system may include an intermediate-level portion of the operating system, while a second, low-level portion of the operating system containing hardware specific aspects are stored in read-only memory. The process of transferring the first portion from non-volatile storage to volatile storage and decompressing the first portion may be initiated by the low-level portion stored in the read-only memory. Once decompressed, the portion of memory storing the first portion of the operating system may be write-protected, and this read-only memory portion may be mapped out of the address space of RAM used by the operating system.

Esfahani, '695, 2:14-25; Esfahani '265, 2:17-29

> Note that in alternative embodiments of the OS 30, some or all of the above mentioned components of the ROM image **53** may be provided as separate, compressed elements. These separate elements may be embodied in separate Boot Info files or in a single Boot Info file. This approach would allow the OS components that are required for a given machine to be individually selected, decompressed as part of the ROM image, and used as part of the OS 30, while unnecessary components could be ignored. For example, components which might be embodied separately include the kernel, the 68K Emulator, particular drivers or other software components that are not common to all Macintosh computers, and code libraries. Hence, this approach may reduce the amount of RAM consumed by the OS. For example, if components are stored in separate boot Info files, then only the necessary Boot Info files

> would need to be read into RAM during the boot process. Alternatively, if components are embodied as discrete components within a single Boot Info file, then the system might read the Boot Info file into RAM, identify the needed components, decompress only the needed components into a separate memory space, and return the unused memory space to the OS for use as system RAM.

Esfahani, ’695, 7:43-65; Esfahani ’265, 7:47-8:2

6.3 wherein the processor is configured:	Esfahani, as evidenced by the example citations below, discloses "wherein the processor is configured:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the processor is configured), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 6 (Preamble) above	

6.4 to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory,	Esfahani, as evidenced by the example citations below, discloses "to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1.1 above.	

6.5 to access the loaded portion of the boot data in the compressed form,	Esfahani, as evidenced by the example citations below, discloses "to access the loaded portion of the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to access the loaded portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1.2 above.	

6.6 to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data, and	Esfahani, as evidenced by the example citations below, discloses “to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1.3 above.	

6.7 to update the boot data list.	Esfahani, as evidenced by the example citations below, discloses "to update the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to update the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1.4.1 above.	

8 (Preamble) A method of loading an operating system for booting a computer system, comprising:	Esfahani, as evidenced by the example citations below, discloses "A method of loading an operating system for booting a computer system, comprising:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method of loading an operating system for booting a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1 (Preamble) above.	

8.1 storing a portion of the operating system in a compressed form in a first memory;	Esfahani, as evidenced by the example citations below, discloses “storing a portion of the operating system in a compressed form in a first memory”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing a portion of the operating system in a compressed form in a first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 6.2 above.	

8.2 loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list;	Esfahani, as evidenced by the example citations below, discloses “loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1.1 above.	

8.3 accessing the loaded portion of the operating system from the second memory in the compressed form;	Esfahani, as evidenced by the example citations below, discloses "accessing the loaded portion of the operating system from the second memory in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded portion of the operating system from the second memory in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1.2 above.	

8.4 decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system;	Esfahani, as evidenced by the example citations below, discloses "decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 1.3 and 1.4.2 above.	

8.5 utilizing the decompressed portion of the operating system to at least partially boot the computer system; and	Esfahani, as evidenced by the example citations below, discloses "utilizing the decompressed portion of the operating system to at least partially boot the computer system"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed portion of the operating system to at least partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

See Claims 1.3 and 1.4.2 above.

See also

> The low-level portion, including hardware-specific code, is stored in a relatively small read-only memory (ROM), while at least part of the intermediate-level portion is stored as a compressed ROM image on a disk or other mass storage device, which may be located remotely from the computer system. Upon power-up or reset of the computer system, the code in the ROM is executed to read the compressed ROM image into random access memory (RAM) of the computer system. The compressed image is then decompressed and executed as part of the boot sequence. Once decompressed, the portion of RAM storing the intermediate-level code is write-protected in the memory map, and the code in boot ROM is deleted from the memory map. Memory space in RAM that is allocated to the intermediate-level code but not used is returned to the operating system for use as part of system RAM.

Esfahani '695 [Abstract]

> A method and apparatus for use in booting a computer system are provided. The method includes loading a compressed image of a first portion of the OS of the computer system into a storage device of the computer system, which may be RAM, for example. The compressed image of the first portion of the OS is then decompressed and executed as part of the boot sequence of the computer system.

Esfahani '695 2:6-12; *also* Esfahani '265, 2:10-17.

> In particular embodiments, the first portion of the operating system may include an intermediate-level portion of the operating system, while a second, low-level portion of the operating system containing hardware specific aspects are stored in read-only memory. The process of transferring the first portion from non-volatile storage to volatile storage and decompressing the first portion may be initiated by the low-level portion stored in the read-only memory. Once decompressed, the portion of memory storing the first portion of the operating system may be write-protected, and this read-only memory portion may be mapped out of the address space of RAM used by the operating system.

Esfahani 695 2:13-25; *also* Esfahani '265, 2:18-29.

> A computer OS using a compressed ROM image in RAM is described. In brief, the low-level portion of an OS of a computer is designed to be separate from the intermediate-level portion of the OS. The low-level portion, which includes hardware-specific code, is stored in a relatively small Boot ROM, while at least part of the intermediate-level portion is stored as a compressed ROM image on a disk or other mass storage device. The mass storage device may be located remotely from the computer system, such as in a file server. Upon power-up or reset of the computer system, the code in the boot ROM is executed to read the compressed ROM image into RAM, i.e., system memory, of the computer system. The compressed image is then decompressed and executed as part of the boot sequence.

Esfahani 695, 2:54-67; *also* Esfahani '265, 2:58-3:4.

> Note that in alternative embodiments of the OS 30, some or all of the above mentioned components of the ROM image 53 may be provided as separate, compressed elements. These separate elements may be embodied in separate Boot Info files or in a single Boot Info file. This approach would allow the OS components that are required for a given machine to be individually selected, decompressed as part of the ROM image, and used as part of the OS 30, while unnecessary components could be ignored.

Esfahani '695, 7:43-51; *also* Esfahani '265, 7:47-55.

> Overall Boot Process
>
> Referring now to FIGS. 5, 6A and 6B, the overall boot process of the improved OS 30 will now be described. In response to power on or reset at block 601 (FIG. 6A), the POST code executes (preliminary diagnostics, boot beep, initialization) at 602. At 603, the Open Firmware

> routine initializes and begins execution, including building the expanded Device Tree and the Interrupt Trees. At 604, the Open Firmware locates the Boot Info file 40, based on defaults and NVRAM settings, and loads the Boot Info file 40 into RAM 12. At 605, the Open Firmware executes the Forth script in the Boot Info file 40, which contains information about the rest of the file (offsets, etc.) and instructions to read both the Trampoline code 52 and the compressed ROM Image 53, and places them into a temporary place in RAM 12. At 606, the Open Firmware transfers control to the Trampoline code 52. At 607, the Trampoline code 52 decompresses the ROM Image 53 in RAM 12. In addition, the Trampoline code reallocates any unused memory space in the ROM Image 52; gathers information about the system from Open Firmware; creates data structures based on this information; terminates Open Firmware, and rearranges the contents of memory to an interim location in physical memory space. At 608, the Trampoline code 52 transfers control to the HardwareInit routine of the (now decompressed) ROM Image 53. At 609, the HardwareInit routine copies data structures to their correct places in memory and then calls the kernel of the OS. Next, at 610 the kernel fills in its data structures and then calls the 68K Emulator. At 611, the 68K Emulator initializes itself and then transfers control to the StartInit routine. At 612, the StartInit routine begins execution, initializing data structures and managers, and booting the MacOS (the high-level portion of the OS). Note that blocks 609 through 612, above, are specific to a Mac OS.

Esfahani ’695, 9:38-10:6; *also* Esfahani ’265, 8:42-9:10.

See also Esfahani, Figs. 1, 4, 5, 6A, and 6B.

8.6 updating the boot data list,	Esfahani, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1.4.1 above.	

8.7 wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form.	Esfahani, as evidenced by the example citations below, discloses "wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 1-1.3 and 5.7 above.	

9.1 The method of claim 8, further comprising: compressing an additional portion of the operating system that is not associated with the boot data list; and	Esfahani, as evidenced by the example citations below, discloses "the method of claim 8, further comprising: compressing an additional portion of the operating system that is not associated with the boot data list"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, compressing an additional portion of the operating system that is not associated with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

See Claim 1.1 above.

See also

> In particular embodiments, the first portion of the operating system may include an intermediate-level portion of the operating system, while a second, low-level portion of the operating system containing hardware specific aspects are stored in read-only memory. The process of transferring the first portion from non-volatile storage to volatile storage and decompressing the first portion may be initiated by the low-level portion stored in the read-only memory. Once decompressed, the portion of memory storing the first portion of the operating system may be write-protected, and this read-only memory portion may be mapped out of the address space of RAM used by the operating system.

Esfahani, '695, 2:14-25; Esfahani '265, 2:17-29

> Note that in alternative embodiments of the OS 30, some or all of the above mentioned components of the ROM image **53** may be provided as separate, compressed elements. These separate elements may be embodied in separate Boot Info files or in a single Boot Info file. This approach would allow the OS components that are required for a given machine to be individually selected, decompressed as part of the ROM image, and used as part of the OS 30, while unnecessary components could be ignored. For example, components which might be embodied separately include the kernel, the 68K Emulator, particular drivers or other software components that are not common to all Macintosh computers, and code libraries. Hence, this approach may reduce the

> amount of RAM consumed by the OS. For example, if components are stored in separate boot Info files, then only the necessary Boot Info files would need to be read into RAM during the boot process. Alternatively, if components are embodied as discrete components within a single Boot Info file, then the system might read the Boot Info file into RAM, identify the needed components, decompress only the needed components into a separate memory space, and return the unused memory space to the OS for use as system RAM.

Esfahani, ’695, 7:43-65; Esfahani ’265, 7:47-8:2

9.2 storing the additional portion of the operating system in the first memory, and	Esfahani, as evidenced by the example citations below, discloses "storing the additional portion of the operating system in the first memory"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing the additional portion of the operating system in the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

See Claim 8.1 above.

See also

> In particular embodiments, the first portion of the operating system may include an intermediate-level portion of the operating system, while a second, low-level portion of the operating system containing hardware specific aspects are stored in read-only memory. The process of transferring the first portion from non-volatile storage to volatile storage and decompressing the first portion may be initiated by the low-level portion stored in the read-only memory. Once decompressed, the portion of memory storing the first portion of the operating system may be write-protected, and this read-only memory portion may be mapped out of the address space of RAM used by the operating system.

Esfahani, '695, 2:14-25; Esfahani '265, 2:17-29

> Note that in alternative embodiments of the OS 30, some or all of the above mentioned components of the ROM image **53** may be provided as separate, compressed elements. These separate elements may be embodied in separate Boot Info files or in a single Boot Info file. This approach would allow the OS components that are required for a given machine to be individually selected, decompressed as part of the ROM image, and used as part of the OS 30, while unnecessary components could be ignored. For example, components which might be embodied separately include the kernel, the 68K Emulator, particular drivers or other software components that are not common to all Macintosh computers, and code libraries. Hence, this approach may reduce the amount of RAM consumed by the OS. For example, if components are stored in separate boot Info files, then only the necessary Boot Info files would need to be read into RAM during the boot process. Alternatively, if components are embodied as discrete components within a single

Boot Info file, then the system might read the Boot Info file into RAM, identify the needed components, decompress only the needed components into a separate memory space, and return the unused memory space to the OS for use as system RAM. Esfahani, '695, 7:43-65; Esfahani '265, 7:47-8:2

9.3 wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system.	Esfahani, as evidenced by the example citations below, discloses "wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

See Claim 8.5 above.

See also

> In particular embodiments, the first portion of the operating system may include an intermediate-level portion of the operating system, while a second, low-level portion of the operating system containing hardware specific aspects are stored in read-only memory. The process of transferring the first portion from non-volatile storage to volatile storage and decompressing the first portion may be initiated by the low-level portion stored in the read-only memory. Once decompressed, the portion of memory storing the first portion of the operating system may be write-protected, and this read-only memory portion may be mapped out of the address space of RAM used by the operating system.

Esfahani, '695, 2:14-25; Esfahani '265, 2:17-29

> Note that in alternative embodiments of the OS 30, some or all of the above mentioned components of the ROM image **53** may be provided as separate, compressed elements. These separate elements may be embodied in separate Boot Info files or in a single Boot Info file. This approach would allow the OS components that are required for a given machine to be individually selected, decompressed as part of the ROM image, and used as part of the OS 30, while unnecessary components could be ignored. For example, components which might be embodied separately include the kernel, the 68K Emulator, particular drivers or other software components that are not common to all Macintosh computers, and code libraries. Hence, this approach may reduce the

> amount of RAM consumed by the OS. For example, if components are stored in separate boot Info files, then only the necessary Boot Info files would need to be read into RAM during the boot process. Alternatively, if components are embodied as discrete components within a single Boot Info file, then the system might read the Boot Info file into RAM, identify the needed components, decompress only the needed components into a separate memory space, and return the unused memory space to the OS for use as system RAM.

Esfahani, '695, 7:43-65; Esfahani '265, 7:47-8:2

10. The method of claim 9, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 9, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

See Claim 9.1 above.

See also

> The ROM Image 53 of the improved OS 30 is similar to the ToolBox ROM code of the earlier Macintosh OS, in that it has a similar layout and contains many of the same components. The image may be compressed using a known compression algorithm, such as LZSS, for example. The ROM Image 53 may also be encoded, if desired.

Esfahani, '695, 8:32-37; Esfahani '265, 8:36-41

Appendix C1
Invalidity of U.S. Patent 8,880,862 based on Esfahani

11 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	Esfahani, as evidenced by the example citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, comprising:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1 (Preamble) above.	

Appendix C1
Invalidity of U.S. Patent 8,880,862 based on Esfahani

11.1 loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system;	Esfahani, as evidenced by the example citations below, discloses “loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1.1 and 6 (Preamble) above.	

11.2 accessing the loaded boot data in compressed form from the memory;	Esfahani, as evidenced by the example citations below, discloses "accessing the loaded boot data in compressed form from the memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded boot data in compressed form from the memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1.2 above.	

11.3 decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form;	Esfahani, as evidenced by the example citations below, discloses "decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1.3 above.	

11.4 utilizing the decompressed boot data to load at least a portion of the operating system for the computer system; and	Esfahani, as evidenced by the example citations below, discloses "utilizing the decompressed boot data to load at least a portion of the operating system for the computer system"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed boot data to load at least a portion of the operating system for the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

> The low-level portion, including hardware-specific code, is stored in a relatively small read-only memory (ROM), while at least part of the intermediate-level portion is stored as a compressed ROM image on a disk or other mass storage device, which may be located remotely from the computer system. Upon power-up or reset of the computer system, the code in the ROM is executed to read the compressed ROM image into random access memory (RAM) of the computer system. The compressed image is then decompressed and executed as part of the boot sequence. Once decompressed, the portion of RAM storing the intermediate-level code is write-protected in the memory map, and the code in boot ROM is deleted from the memory map. Memory space in RAM that is allocated to the intermediate-level code but not used is returned to the operating system for use as part of system RAM.

Esfahani '695 [Abstract]

> A method and apparatus for use in booting a computer system are provided. The method includes loading a compressed image of a first portion of the OS of the computer system into a storage device of the computer system, which may be RAM, for example. The compressed image of the first portion of the OS is then decompressed and executed as part of the boot sequence of the computer system.

Esfahani '695 2:6-12; *also* Esfahani '265, 2:10-17.

> In particular embodiments, the first portion of the operating system may include an intermediate-level portion of the operating system, while a second, low-level portion of the operating system containing hardware specific aspects are stored in read-only memory. The process of transferring the first portion from non-volatile storage to volatile storage

> and decompressing the first portion may be initiated by the low-level portion stored in the read-only memory. Once decompressed, the portion of memory storing the first portion of the operating system may be write-protected, and this read-only memory portion may be mapped out of the address space of RAM used by the operating system.

Esfahani 695 2:13-25; *also* Esfahani '265, 2:18-29.

> A computer OS using a compressed ROM image in RAM is described. In brief, the low-level portion of an OS of a computer is designed to be separate from the intermediate-level portion of the OS. The low-level portion, which includes hardware-specific code, is stored in a relatively small Boot ROM, while at least part of the intermediate-level portion is stored as a compressed ROM image on a disk or other mass storage device. The mass storage device may be located remotely from the computer system, such as in a file server. Upon power-up or reset of the computer system, the code in the boot ROM is executed to read the compressed ROM image into RAM, i.e., system memory, of the computer system. The compressed image is then decompressed and executed as part of the boot sequence.

Esfahani 695, 2:54-67; *also* Esfahani '265, 2:58-3:4.

> Note that in alternative embodiments of the OS 30, some or all of the above mentioned components of the ROM image 53 may be provided as separate, compressed elements. These separate elements may be embodied in separate Boot Info files or in a single Boot Info file. This approach would allow the OS components that are required for a given machine to be individually selected, decompressed as part of the ROM image, and used as part of the OS 30, while unnecessary components could be ignored.

Esfahani '695, 7:43-51; *also* Esfahani '265, 7:47-55.

> Overall Boot Process
>
> Referring now to FIGS. 5, 6A and 6B, the overall boot process of the improved OS 30 will now be described. In response to power on or reset at block 601 (FIG. 6A), the POST code executes (preliminary diagnostics, boot beep, initialization) at 602. At 603, the Open Firmware routine initializes and begins execution, including building the expanded Device Tree and the Interrupt Trees. At 604, the Open Firmware locates the Boot Info file 40, based on defaults and NVRAM settings, and loads the Boot Info file 40 into RAM 12. At 605, the Open Firmware executes the Forth script in the Boot Info file 40, which

> contains information about the rest of the file (offsets, etc.) and instructions to read both the Trampoline code 52 and the compressed ROM Image 53, and places them into a temporary place in RAM 12. At 606, the Open Firmware transfers control to the Trampoline code 52. At 607, the Trampoline code 52 decompresses the ROM Image 53 in RAM 12. In addition, the Trampoline code reallocates any unused memory space in the ROM Image 52; gathers information about the system from Open Firmware; creates data structures based on this information; terminates Open Firmware, and rearranges the contents of memory to an interim location in physical memory space. At 608, the Trampoline code 52 transfers control to the HardwareInit routine of the (now decompressed) ROM Image 53. At 609, the HardwareInit routine copies data structures to their correct places in memory and then calls the kernel of the OS. Next, at 610 the kernel fills in its data structures and then calls the 68K Emulator. At 611, the 68K Emulator initializes itself and then transfers control to the StartInit routine. At 612, the StartInit routine begins execution, initializing data structures and managers, and booting the MacOS (the high-level portion of the OS). Note that blocks 609 through 612, above, are specific to a Mac OS.

Esfahani ’695, 9:38-10:6; *also* Esfahani ’265, 8:42-9:10.

See also Esfahani, Figs. 1, 4, 5, 6A, and 6B.

11.5 updating the boot data list.	Esfahani, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1.4.1 above.	

Appendix C1
Invalidity of U.S. Patent 8,880,862 based on Esfahani

13 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	Esfahani, as evidenced by the example citations below, discloses “a method for providing accelerated loading of an operating system in a computer system, comprising”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1 (Preamble) above.	

13.1 loading boot data in a compressed form that is associated with a boot data list from a boot device;	Esfahani, as evidenced by the example citations below, discloses "loading boot data in a compressed form that is associated with a boot data list from a boot device"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading boot data in a compressed form that is associated with a boot data list from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1.1 above.	

13.2 accessing the loaded boot data in the compressed form;	Esfahani, as evidenced by the example citations below, discloses "accessing the loaded boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1.2 above.	

13.3 decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form;	Esfahani, as evidenced by the example citations below, discloses "decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1.3 above.	

13.4 updating the boot data list.	Esfahani, as evidenced by the example citations below, discloses “updating the boot data list”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1.4.1 above.	

Appendix C1
Invalidity of U.S. Patent 8,880,862 based on Esfahani

14 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	Esfahani, as evidenced by the example citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, comprising"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1 (Preamble) above.	

14.1 accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list;	Esfahani, as evidenced by the example citations below, discloses "accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

See Claims 1.1 and 1.2 above.

See also

> During start-up, the Boot Info file 40 is loaded into RAM 12, and the compressed mid-level OS 32 is decompressed. Hence, the mid-level OS 32 is essentially a compressed ROM image. The mid-level OS 32 is inserted into the memory map of the computer system as if it were firmware in ROM. That is, the ROM image can be write-protected in the memory map.

Esfahani '695, 5:44-51; *also* Esfahani '265, 5:48-54.

> Referring now to FIG. 5, the low-level OS portion 31 and the Boot Info file 40 are illustrated in greater detail. The low-level portion 31 stored in Boot ROM 11 contains the code needed to start up the computer, initialize and examine the hardware, provide a Device Tree (per Open Firmware) to describe the hardware, provide hardware access services, and transfer control to the OS. In one embodiment, the components of the low-level portion 31 include:
>
>> code for performing Power-On Self Test (POST), including code for performing diagnostics, generating a boot beep and an error beep;
>>
>> Open Firmware code;
>>
>> hardware-specific Mac OS drivers ("ndrv's") that are needed at boot time (drivers needed at boot time, e.g.,video drivers,

> > network drivers, or disk drivers, are loaded from the Device Tree);
> >
> > HardwareInit code (i.e., the lowest-level code for initializing the CPU, RAM, clock, system bus, etc.) without Mac OS-specific code;
> >
> > code for performing Run-Time Abstraction Services (RTAS). Certain hardware devices differ from machine to machine, but provide similar functions. RTAS provides such functions, including functions for accessing the real-time clock, NVRAM 20, restart, shutdown, and PCI configuration cycles. The I/O primitives for these functions in the ROM Image make use of RTAS.

Esfahani ’695, 6:29-54; *also* Esfahani ’265, 6:31-57.

> The Boot Info file 40 will now be described in greater detail. The Boot Info file 40 may be stored in the System Folder of the start-up volume. Alternatively, the Boot Info file 40 may be provided by a network server using, for example, the Bootstrap Protocol (BootP), which is described in B. Croft et al., Network Working Group Request for Comments (RFC) 951, September 1985. Referring to FIG. 5, the Boot Info file 40 includes Open Firmware-specific Mac OS code 52, referred to as the “Trampoline code”; an Open Firmware header 51, which includes a Forth script that performs operations necessary to the start-up of the OS, including validation tests and transfer of control to the Trampoline code; and a compressed ROM Image 53, which represents the mid-level portion 32 of the OS 30. The purpose of the header 51 is, generally, to specify the locations of the other components of the Boot Info file 40. The purpose of the Trampoline code 53 generally is to handle the transition between the Open Firmware code in the Boot Rom 11 and the ROM Image 52, as will be described in greater detail below.

Esfahani ’695, 7:5-24; *also* Esfahani ’265, 7:9:28.

> The Boot Info file 40 resides on the boot device (e.g., a disk, or on a network) and has a localizable name.

Esfahani ’695, 7:66-67; *also* Esfahani ’265, 8:3-4.

See also Esfahani, Figs. 1, 4, 5, 6A, and 6B.

14.2 loading the boot data into a memory; and	Esfahani, as evidenced by the example citations below, discloses “loading the boot data into a memory”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the boot data into a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1.1 above.	

14.3 servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form; and	Esfahani, as evidenced by the example citations below, discloses "servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

See Claims 1.2 and 1.3 above.

See also

> The low-level portion, including hardware-specific code, is stored in a relatively small read-only memory (ROM), while at least part of the intermediate-level portion is stored as a compressed ROM image on a disk or other mass storage device, which may be located remotely from the computer system. Upon power-up or reset of the computer system, the code in the ROM is executed to read the compressed ROM image into random access memory (RAM) of the computer system. The compressed image is then decompressed and executed as part of the boot sequence. Once decompressed, the portion of RAM storing the intermediate-level code is write-protected in the memory map, and the code in boot ROM is deleted from the memory map. Memory space in RAM that is allocated to the intermediate-level code but not used is returned to the operating system for use as part of system RAM.

Esfahani '695 [Abstract]

> A method and apparatus for use in booting a computer system are provided. The method includes loading a compressed image of a first

> portion of the OS of the computer system into a storage device of the computer system, which may be RAM, for example. The compressed image of the first portion of the OS is then decompressed and executed as part of the boot sequence of the computer system.

Esfahani ’695 2:6-12; *also* Esfahani ’265, 2:10-17.

> In particular embodiments, the first portion of the operating system may include an intermediate-level portion of the operating system, while a second, low-level portion of the operating system containing hardware specific aspects are stored in read-only memory. The process of transferring the first portion from non-volatile storage to volatile storage and decompressing the first portion may be initiated by the low-level portion stored in the read-only memory. Once decompressed, the portion of memory storing the first portion of the operating system may be write-protected, and this read-only memory portion may be mapped out of the address space of RAM used by the operating system.

Esfahani 695 2:13-25; *also* Esfahani ’265, 2:18-29.

> A computer OS using a compressed ROM image in RAM is described. In brief, the low-level portion of an OS of a computer is designed to be separate from the intermediate-level portion of the OS. The low-level portion, which includes hardware-specific code, is stored in a relatively small Boot ROM, while at least part of the intermediate-level portion is stored as a compressed ROM image on a disk or other mass storage device. The mass storage device may be located remotely from the computer system, such as in a file server. Upon power-up or reset of the computer system, the code in the boot ROM is executed to read the compressed ROM image into RAM, i.e., system memory, of the computer system. The compressed image is then decompressed and executed as part of the boot sequence.

Esfahani 695, 2:54-67; *also* Esfahani ’265, 2:58-3:4.

> Referring now to FIG. 4, in the improved OS, the low-level (hardware-specific) OS code 31 resides in firmware, in order to handle start-up activities of the computer system. This code fits into one, relatively small ROM, referred to as the Boot ROM 11. Thus, Boot ROM 11 includes all of the hardware-specific code and tables needed to start up the computer as well as to boot the OS and provide common hardware access services the OS might require. Note that the Boot ROM code is not specific to the MacOS or to any other OS. All higher-level software resides elsewhere, as will now be described.

> Prior to start-up, the mid-level portion 32 of OS 30 (which corresponds to part of the ToolBox ROM of earlier Macintosh computers) resides in compressed form in a file 40, referred to as Boot Info file 40.

Esfahani ’695, 5:7-23; *also* Esfahani ’265, 5:10-24.

> During start-up, the Boot Info file 40 is loaded into RAM 12, and the compressed mid-level OS 32 is decompressed. Hence, the mid-level OS 32 is essentially a compressed ROM image. The mid-level OS 32 is inserted into the memory map of the computer system as if it were firmware in ROM. That is, the ROM image can be write-protected in the memory map.

Esfahani ’695, 5:44-51; *also* Esfahani ’265, 5:48-54.

> Referring now to FIG. 5, the low-level OS portion 31 and the Boot Info file 40 are illustrated in greater detail. The low-level portion 31 stored in Boot ROM 11 contains the code needed to start up the computer, initialize and examine the hardware, provide a Device Tree (per Open Firmware) to describe the hardware, provide hardware access services, and transfer control to the OS. In one embodiment, the components of the low-level portion 31 include:
>
> > code for performing Power-On Self Test (POST), including code for performing diagnostics, generating a boot beep and an error beep;
> >
> > Open Firmware code;
> >
> > hardware-specific Mac OS drivers (“ndrv's”) that are needed at boot time (drivers needed at boot time, e.g.,video drivers, network drivers, or disk drivers, are loaded from the Device Tree);
> >
> > HardwareInit code (i.e., the lowest-level code for initializing the CPU, RAM, clock, system bus, etc.) without Mac OS-specific code;
> >
> > code for performing Run-Time Abstraction Services (RTAS). Certain hardware devices differ from machine to machine, but provide similar functions. RTAS provides such functions, including functions for accessing the real-time clock, NVRAM 20, restart, shutdown, and PCI configuration cycles. The I/O

> primitives for these functions in the ROM Image make use of RTAS.

Esfahani '695, 6:29-54; *also* Esfahani '265, 6:31-57.

> The Boot Info file 40 will now be described in greater detail. The Boot Info file 40 may be stored in the System Folder of the start-up volume. Alternatively, the Boot Info file 40 may be provided by a network server using, for example, the Bootstrap Protocol (BootP), which is described in B. Croft et al., Network Working Group Request for Comments (RFC) 951, September 1985. Referring to FIG. 5, the Boot Info file 40 includes Open Firmware-specific Mac OS code 52, referred to as the "Trampoline code"; an Open Firmware header 51, which includes a Forth script that performs operations necessary to the start-up of the OS, including validation tests and transfer of control to the Trampoline code; and a compressed ROM Image 53, which represents the mid-level portion 32 of the OS 30. The purpose of the header 51 is, generally, to specify the locations of the other components of the Boot Info file 40. The purpose of the Trampoline code 53 generally is to handle the transition between the Open Firmware code in the Boot Rom 11 and the ROM Image 52, as will be described in greater detail below.

Esfahani '695, 7:5-24; *also* Esfahani '265, 7:9:28.

> Note that in alternative embodiments of the OS 30, some or all of the above mentioned components of the ROM image 53 may be provided as separate, compressed elements. These separate elements may be embodied in separate Boot Info files or in a single Boot Info file. This approach would allow the OS components that are required for a given machine to be individually selected, decompressed as part of the ROM image, and used as part of the OS 30, while unnecessary components could be ignored.

Esfahani '695, 7:43-51; *also* Esfahani '265, 7:47-55.

> The Boot Info file 40 resides on the boot device (e.g., a disk, or on a network) and has a localizable name.

Esfahani '695, 7:66-67; *also* Esfahani '265, 8:3-4.

> Overall Boot Process
>
> Referring now to FIGS. 5, 6A and 6B, the overall boot process of the improved OS 30 will now be described. In response to power on or reset

> at block 601 (FIG. 6A), the POST code executes (preliminary diagnostics, boot beep, initialization) at 602. At 603, the Open Firmware routine initializes and begins execution, including building the expanded Device Tree and the Interrupt Trees. At 604, the Open Firmware locates the Boot Info file 40, based on defaults and NVRAM settings, and loads the Boot Info file 40 into RAM 12. At 605, the Open Firmware executes the Forth script in the Boot Info file 40, which contains information about the rest of the file (offsets, etc.) and instructions to read both the Trampoline code 52 and the compressed ROM Image 53, and places them into a temporary place in RAM 12. At 606, the Open Firmware transfers control to the Trampoline code 52. At 607, the Trampoline code 52 decompresses the ROM Image 53 in RAM 12. In addition, the Trampoline code reallocates any unused memory space in the ROM Image 52; gathers information about the system from Open Firmware; creates data structures based on this information; terminates Open Firmware, and rearranges the contents of memory to an interim location in physical memory space. At 608, the Trampoline code 52 transfers control to the HardwareInit routine of the (now decompressed) ROM Image 53. At 609, the HardwareInit routine copies data structures to their correct places in memory and then calls the kernel of the OS. Next, at 610 the kernel fills in its data structures and then calls the 68K Emulator. At 611, the 68K Emulator initializes itself and then transfers control to the StartInit routine. At 612, the StartInit routine begins execution, initializing data structures and managers, and booting the MacOS (the high-level portion of the OS). Note that blocks 609 through 612, above, are specific to a Mac OS.

Esfahani ’695, 9:38-10:6; *also* Esfahani ’265, 8:42-9:10.

See also Esfahani, Figs. 1, 4, 5, 6A, and 6B.

14.4 updating the boot data list.	Esfahani, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1.4.1 above.	

15. The method of claim 14, wherein the boot data comprises: a program code associated with the operating system.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 14, wherein the boot data comprises: a program code associated with the operating system."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the boot data comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

See Claims 1 (Preamble) and 1.1 above.

See also

> Refer now to FIG. 2, which illustrates the structure of a traditional Macintosh OS. The OS 22 provides an interface between the hardware 23 of the computer system and the software applications 21. The OS includes the so-called MacOS 22A, which is the high-level portion of the OS 22. In this context, "high-level" refers to the portion of the OS 22 which is least hardware-specific and has the greatest degree of abstraction. Traditionally, this file would reside on disk or other mass storage device and would typically be the last portion of the OS22 to be invoked during the boot process. The (traditional Macintosh) OS22 also includes the so-called ToolBox ROM code 22B. The ToolBox ROM code 22B is firmware that resides in a ROM. The ToolBox ROM code 22B represents the middle- or intermediate-level and low-level portions of the OS22. In this context, "low-level" refers to the portion of the OS which is most hardware-specific and has the smallest degree of abstraction. The middle-level portion has degrees of hardware-dependence and abstraction lower than those of the high-level OS22A and greater than those of the low-level OS.

Esfahani, '695, 4:1-20; Esfahani '265, 4:5-25

See also Esfahani Fig. 2

Appendix C1
Invalidity of U.S. Patent 8,880,862 based on Esfahani

16. The method of claim 14, wherein the operating system comprises: a plurality of files.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 14, wherein the operating system comprises: a plurality of files."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

See Claim 1 and 14 above

See also

> In particular embodiments, the first portion of the operating system may include an intermediate-level portion of the operating system, while a second, low-level portion of the operating system containing hardware specific aspects are stored in read-only memory. The process of transferring the first portion from non-volatile storage to volatile storage and decompressing the first portion may be initiated by the low-level portion stored in the read-only memory. Once decompressed, the portion of memory storing the first portion of the operating system may be write-protected, and this read-only memory portion may be mapped out of the address space of RAM used by the operating system.

Esfahani, '695, 2:14-25; Esfahani '265, 2:17-29

> Note that in alternative embodiments of the OS 30, some or all of the above mentioned components of the ROM image **53** may be provided as separate, compressed elements. These separate elements may be embodied in separate Boot Info files or in a single Boot Info file. This approach would allow the OS components that are required for a given machine to be individually selected, decompressed as part of the ROM image, and used as part of the OS 30, while unnecessary components could be ignored. For example, components which might be embodied separately include the kernel, the 68K Emulator, particular drivers or other software components that are not common to all Macintosh computers, and code libraries. Hence, this approach may reduce the amount of RAM consumed by the OS. For example, if components are stored in separate boot Info files, then only the necessary Boot Info files would need to be read into RAM during the boot process. Alternatively,

if components are embodied as discrete components within a single Boot Info file, then the system might read the Boot Info file into RAM, identify the needed components, decompress only the needed components into a separate memory space, and return the unused memory space to the OS for use as system RAM. Esfahani, ’695, 7:43-65; Esfahani ’265, 7:47-8:2

17. The method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

See Claim 15 above.

See also

> Refer now to FIG. 2, which illustrates the structure of a traditional Macintosh OS. The OS 22 provides an interface between the hardware 23 of the computer system and the software applications 21. The OS includes the so-called MacOS 22A, which is the high-level portion of the OS 22. In this context, "high-level" refers to the portion of the OS 22 which is least hardware-specific and has the greatest degree of abstraction. Traditionally, this file would reside on disk or other mass storage device and would typically be the last portion of the OS22 to be invoked during the boot process. The (traditional Macintosh) OS22 also includes the so-called ToolBox ROM code 22B. The ToolBox ROM code 22B is firmware that resides in a ROM. The ToolBox ROM code 22B represents the middle- or intermediate-level and low-level portions of the OS22. In this context, "low-level" refers to the portion of the OS which is most hardware-specific and has the smallest degree of abstraction. The middle-level portion has degrees of hardware-dependence and abstraction lower than those of the high-level OS22A and greater than those of the low-level OS.

Esfahani, '695, 4:1-20; Esfahani '265, 4:5-25

See also Esfahani Fig. 2

Appendix C1
Invalidity of U.S. Patent 8,880,862 based on Esfahani

19.1 The method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:	Esfahani, as evidenced by the example citations below, discloses "the method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 1.1 and 1.2 above	

19.2 associating the accessed boot data that is not associated with the boot data list to the boot data list.	Esfahani, as evidenced by the example citations below, discloses "associating the accessed boot data that is not associated with the boot data list to the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, associating the accessed boot data that is not associated with the boot data list to the boot data list.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 2 above.	

23. The method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files.	Esfahani, as evidenced by the example citations below, discloses “the method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 16 above.	

24. The method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system.
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 15 above.	

27. The method of claim 1, wherein the memory comprises: a physical memory.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 1, wherein the memory comprises: a physical memory."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

See Claims 1.1 and 1.2 above

See also

> In particular embodiments, the first portion of the operating system may include an intermediate-level portion of the operating system, while a second, low-level portion of the operating system containing hardware specific aspects are stored in read-only memory. The process of transferring the first portion from non-volatile storage to volatile storage and decompressing the first portion may be initiated by the low-level portion stored in the read-only memory. Once decompressed, the portion of memory storing the first portion of the operating system may be write-protected, and this read-only memory portion may be mapped out of the address space of RAM used by the operating system.

Esfahani, '695, 2:14-25; Esfahani '265, 2:17-29

> FIG. 1 illustrates a computer system 1 in which the present invention may be implemented. Note that while FIG. 1 illustrates the major components of a computer system, it is not intended to represent any particular architecture or manner of interconnecting the component; such details are not germane to the present invention. The computer system of FIG. 1 may be, for example, an Apple Macintosh computer, such as an Apple iMac computer. As shown, the computer system 1 of FIG. 1 includes a microprocessor 10, a read-only memory (ROM) 11, random access memory (RAM) 12, each connected to a bus system 18. The bus system 18 may include one or more buses connected to each other through various bridges, controllers and/or adapters, such as are well-known in the art. For example, the bus system may include a "system bus" that is connected through an adapter to one or more expansion buses, such as a Peripheral Component Interconnect (PCI) bus, or the like. Also coupled to the bus system 18 are a mass storage

> device 13, a display device 14, a keyboard 15, a pointing device 16, a communication device 17, and non-volatile RAM (NVRAM) 20. A cache memory 19 is coupled to the microprocessor 10.

Esfahani, ’695, 3:1-22; Esfahani ’265, 3:4-26

> The Boot Info file 40 resides on the boot device (e.g., a disk, or on a network) and has a localizable name. Identification information that leads to the file's path may be stored in NVRAM 20 or another suitable location, and the search algorithm for a usable Boot Info file parallels the search mechanism across SCSI, ATA, etc., used in the earlier Macintosh OS's StartSearch routine. By default, the Boot Info file 40 is located by using the current, active System folder's dirID (directory ID) in the boot block of each partition of the Hierarchical File System (HFS) and then searching for a file with a predetermined file type. Searching by file type is done to allow localization of the file.

Esfahani, ’695, 7:66-8:10; Esfahani ’265, 8:3-14

See also Esfahani Fig. 1

28. The method of claim 1, wherein the operating system comprises: a plurality of files.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 1, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 16 above.	

29. The method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 15 and 17 above.	

31. The method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

See Claims 1.1 and 1.2 above

See also

> In particular embodiments, the first portion of the operating system may include an intermediate-level portion of the operating system, while a second, low-level portion of the operating system containing hardware specific aspects are stored in read-only memory. The process of transferring the first portion from non-volatile storage to volatile storage and decompressing the first portion may be initiated by the low-level portion stored in the read-only memory. Once decompressed, the portion of memory storing the first portion of the operating system may be write-protected, and this read-only memory portion may be mapped out of the address space of RAM used by the operating system.

Esfahani, '695, 2:14-25; Esfahani '265, 2:17-29

> FIG. 1 illustrates a computer system 1 in which the present invention may be implemented. Note that while FIG. 1 illustrates the major components of a computer system, it is not intended to represent any particular architecture or manner of interconnecting the component; such details are not germane to the present invention. The computer system of FIG. 1 may be, for example, an Apple Macintosh computer, such as an Apple iMac computer. As shown, the computer system 1 of FIG. 1 includes a microprocessor 10, a read-only memory (ROM) 11, random access memory (RAM) 12, each connected to a bus system 18. The bus system 18 may include one or more buses connected to each other through various bridges, controllers and/or adapters, such as are well-known in the art. For example, the bus system may include a

> "system bus" that is connected through an adapter to one or more expansion buses, such as a Peripheral Component Interconnect (PCI) bus, or the like. Also coupled to the bus system 18 are a mass storage device 13, a display device 14, a keyboard 15, a pointing device 16, a communication device 17, and non-volatile RAM (NVRAM) 20. A cache memory 19 is coupled to the microprocessor 10.

Esfahani, '695, 3:1-22; Esfahani '265, 3:4-26

> The Boot Info file 40 resides on the boot device (e.g., a disk, or on a network) and has a localizable name. Identification information that leads to the file's path may be stored in NVRAM 20 or another suitable location, and the search algorithm for a usable Boot Info file parallels the search mechanism across SCSI, ATA, etc., used in the earlier Macintosh OS's StartSearch routine. By default, the Boot Info file 40 is located by using the current, active System folder's dirID (directory ID) in the boot block of each partition of the Hierarchical File System (HFS) and then searching for a file with a predetermined file type. Searching by file type is done to allow localization of the file.

Esfahani, '695, 7:66-8:10; Esfahani '265, 8:3-14

See also Esfahani Fig. 1

32. The method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 1.1 above *See also* The ROM Image 53 of the improved OS 30 is similar to the ToolBox ROM code of the earlier Macintosh OS, in that it has a similar layout and contains many of the same components. The image may be compressed using a known compression algorithm, such as LZSS, for example. The ROM Image 53 may also be encoded, if desired. Esfahani, '695, 8:32-37; Esfahani '265, 8:36-41	

33. The method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 1.1 and 32 above. *See also* > The ROM Image 53 of the improved OS 30 is similar to the ToolBox ROM code of the earlier Macintosh OS, in that it has a similar layout and contains many of the same components. The image may be compressed using a known compression algorithm, such as LZSS, for example. The ROM Image 53 may also be encoded, if desired. Esfahani, '695, 8:32-37; Esfahani '265, 8:36-41	

35. The method of claim 5, wherein the compressed boot data represents a plurality of files.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 5, wherein the compressed boot data represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the compressed boot data represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 16 and 23 above.	

Appendix C1
Invalidity of U.S. Patent 8,880,862 based on Esfahani

36. The method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 15 and 17 above.	

39. The method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 1.1 and 1.2 above	

40. The method of claim 36, wherein the operating system comprises: a plurality of files.	Esfahani, as evidenced by the example citations below, discloses “the method of claim 36, wherein the operating system comprises: a plurality of files”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 36, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 16 and 23 above.	

43. The method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 31 above.	

44. The method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 32 above.	

45. The method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 32 and 33 above.	

47. The system of claim 6, wherein the boot data in the compressed form represents a plurality of files.	Esfahani, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 16 and 23 above.	

48. The system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system.	Esfahani, as evidenced by the example citations below, discloses “the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 15, 17 and 24 above.	

49. The system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form.	Esfahani, as evidenced by the example citations below, discloses "the system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 10 above.	

50. The system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form.	Esfahani, as evidenced by the example citations below, discloses "the system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

> The low-level portion, including hardware-specific code, is stored in a relatively small read-only memory (ROM), while at least part of the intermediate-level portion is stored as a compressed ROM image on a disk or other mass storage device, which may be located remotely from the computer system. Upon power-up or reset of the computer system, the code in the ROM is executed to read the compressed ROM image into random access memory (RAM) of the computer system. The compressed image is then decompressed and executed as part of the boot sequence. Once decompressed, the portion of RAM storing the intermediate-level code is write-protected in the memory map, and the code in boot ROM is deleted from the memory map. Memory space in RAM that is allocated to the intermediate-level code but not used is returned to the operating system for use as part of system RAM.

Esfahani '695 [Abstract]

> A method and apparatus for use in booting a computer system are provided. The method includes loading a compressed image of a first portion of the OS of the computer system into a storage device of the computer system, which may be RAM, for example. The compressed image of the first portion of the OS is then decompressed and executed as part of the boot sequence of the computer system.

Esfahani '695 2:6-12; *also* Esfahani '265, 2:10-17.

> In particular embodiments, the first portion of the operating system may include an intermediate-level portion of the operating system, while a second, low-level portion of the operating system containing hardware specific aspects are stored in read-only memory. The process of

> transferring the first portion from non-volatile storage to volatile storage and decompressing the first portion may be initiated by the low-level portion stored in the read-only memory. Once decompressed, the portion of memory storing the first portion of the operating system may be write-protected, and this read-only memory portion may be mapped out of the address space of RAM used by the operating system.

Esfahani 695 2:13-25; *also* Esfahani '265, 2:18-29.

> A computer OS using a compressed ROM image in RAM is described. In brief, the low-level portion of an OS of a computer is designed to be separate from the intermediate-level portion of the OS. The low-level portion, which includes hardware-specific code, is stored in a relatively small Boot ROM, while at least part of the intermediate-level portion is stored as a compressed ROM image on a disk or other mass storage device. The mass storage device may be located remotely from the computer system, such as in a file server. Upon power-up or reset of the computer system, the code in the boot ROM is executed to read the compressed ROM image into RAM, i.e., system memory, of the computer system. The compressed image is then decompressed and executed as part of the boot sequence.

Esfahani 695, 2:54-67; *also* Esfahani '265, 2:58-3:4.

> Referring now to FIG. 4, in the improved OS, the low-level (hardware-specific) OS code 31 resides in firmware, in order to handle start-up activities of the computer system. This code fits into one, relatively small ROM, referred to as the Boot ROM 11. Thus, Boot ROM 11 includes all of the hardware-specific code and tables needed to start up the computer as well as to boot the OS and provide common hardware access services the OS might require. Note that the Boot ROM code is not specific to the MacOS or to any other OS. All higher-level software resides elsewhere, as will now be described.
>
> Prior to start-up, the mid-level portion 32 of OS 30 (which corresponds to part of the ToolBox ROM of earlier Macintosh computers) resides in compressed form in a file 40, referred to as Boot Info file 40.

Esfahani '695, 5:7-23; *also* Esfahani '265, 5:10-24.

> During start-up, the Boot Info file 40 is loaded into RAM 12, and the compressed mid-level OS 32 is decompressed. Hence, the mid-level OS 32 is essentially a compressed ROM image. The mid-level OS 32 is inserted into the memory map of the computer system as if it were

firmware in ROM. That is, the ROM image can be write-protected in the memory map.

Esfahani ’695, 5:44-51; *also* Esfahani ’265, 5:48-54.

Referring now to FIG. 5, the low-level OS portion 31 and the Boot Info file 40 are illustrated in greater detail. The low-level portion 31 stored in Boot ROM 11 contains the code needed to start up the computer, initialize and examine the hardware, provide a Device Tree (per Open Firmware) to describe the hardware, provide hardware access services, and transfer control to the OS. In one embodiment, the components of the low-level portion 31 include:

code for performing Power-On Self Test (POST), including code for performing diagnostics, generating a boot beep and an error beep;

Open Firmware code;

hardware-specific Mac OS drivers (“ndrv's”) that are needed at boot time (drivers needed at boot time, e.g.,video drivers, network drivers, or disk drivers, are loaded from the Device Tree);

HardwareInit code (i.e., the lowest-level code for initializing the CPU, RAM, clock, system bus, etc.) without Mac OS-specific code;

code for performing Run-Time Abstraction Services (RTAS). Certain hardware devices differ from machine to machine, but provide similar functions. RTAS provides such functions, including functions for accessing the real-time clock, NVRAM 20, restart, shutdown, and PCI configuration cycles. The I/O primitives for these functions in the ROM Image make use of RTAS.

Esfahani ’695, 6:29-54; *also* Esfahani ’265, 6:31-57.

Note that in alternative embodiments of the OS 30, some or all of the above mentioned components of the ROM image 53 may be provided as separate, compressed elements. These separate elements may be embodied in separate Boot Info files or in a single Boot Info file. This approach would allow the OS components that are required for a given machine to be individually selected, decompressed as part of the ROM

image, and used as part of the OS 30, while unnecessary components could be ignored. Esfahani ’695, 7:43-51; *also* Esfahani ’265, 7:47-55. *See also* Esfahani, Figs. 1, 4, 5, 6A, and 6B.

51. The system of claim 6, wherein the first memory comprises: a physical memory.	Esfahani, as evidenced by the example citations below, discloses "the system of claim 6, wherein the first memory comprises: a physical memory"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the first memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

See Claims 27 and 39 above.

52. The system of claim 6, wherein the boot data the compressed form comprises: a plurality of files.	Esfahani, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data the compressed form comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data the compressed form comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 16 and 23 above.	

53. The system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program.	Esfahani, as evidenced by the example citations below, discloses “the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 15, 17 and 24 above.	

59. The method of claim 8, wherein the operating system in the compressed form represents a plurality of files.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 16 and 23 above.	

60. The method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 15, 17 and 24 above.	

63. The method of claim 8, wherein the second memory comprises: a physical memory.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 8, wherein the second memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the second memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 27 and 38 above.	

64. The method of claim 8, wherein the operating system comprises: a plurality of files.	Esfahani, as evidenced by the example citations below, discloses “the method of claim 8, wherein the operating system comprises: a plurality of files”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 16 and 23 above.	

65. The method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 15, 17 and 24 above.	

67. The method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 31 above.	

71. The method of claim 11, wherein the boot data in the compressed form represents a plurality of files.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 11, wherein the boot data in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 16 and 23 above.	

72. The method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 15, 17 and 24 above.	

75. The method of claim 11, wherein the memory comprises: a physical memory.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 11, wherein the memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 27 above.	

76. The method of claim 11, wherein the operating system comprises: a plurality of files.	Esfahani, as evidenced by the example citations below, discloses “the method of claim 11, wherein the operating system comprises: a plurality of files”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 16 and 23 above.	

77. The method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 15, 17 and 24 above.	

79. The method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 31 above.	

80. The method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 32, 33 and 49 above.	

81. The method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 32, 33 and 49 above.	

83. The method of claim 13, wherein the boot data in the compressed form represents a plurality of files.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 13, wherein the boot data in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 16 and 23 above.	

84. The method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system.	Esfahani, as evidenced by the example citations below, discloses “the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 15, 17 and 24 above.	

87. The method of claim 13, wherein the memory comprises: a physical memory.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 13, wherein the memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 27 above.	

88. The method of claim 13, wherein the operating system comprises: a plurality of files.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 13, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 16 and 23 above.	

89. The method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 15, 17 and 24 above.	

91. The method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 31 above.	

92. The method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 32, 33 and 49 above.	

93. The method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data.	Esfahani, as evidenced by the example citations below, discloses “the method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Esfahani discloses this limitation: *See* Claim 32, 33, and 49 above.	

97.1 The method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list,	Esfahani, as evidenced by the example citations below, discloses "the method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 9.1-9.3 and 19 above.	

97.2 and wherein the updating comprises: associating the additional compressed boot data with the boot data list.	Esfahani, as evidenced by the example citations below, discloses "and wherein the updating comprises: associating the additional compressed boot data with the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, and wherein the updating comprises: associating the additional compressed boot data with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 1.4.1, 2, 4, 5.6, and 8.6 above.	

98. The method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 1.4.1, 5.6, and 8.6 above.	

107. The method of claim 2, further comprising: storing the updated boot list in a non-volatile memory.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 2, further comprising: storing the updated boot list in a non-volatile memory."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 2, further comprising: storing the updated boot list in a non-volatile memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Esfahani discloses this limitation:

See Claims 1.4.1, 5.6, and 8.6 above.

See also

> In particular embodiments, the first portion of the operating system may include an intermediate-level portion of the operating system, while a second, low-level portion of the operating system containing hardware specific aspects are stored in read-only memory. The process of transferring the first portion from non-volatile storage to volatile storage and decompressing the first portion may be initiated by the low-level portion stored in the read-only memory. Once decompressed, the portion of memory storing the first portion of the operating system may be write-protected, and this read-only memory portion may be mapped out of the address space of RAM used by the operating system.

Esfahani, '695, 2:14-25; Esfahani '265, 2:17-29

> FIG. 1 illustrates a computer system 1 in which the present invention may be implemented. Note that while FIG. 1 illustrates the major components of a computer system, it is not intended to represent any particular architecture or manner of interconnecting the component; such details are not germane to the present invention. The computer system of FIG. 1 may be, for example, an Apple Macintosh computer, such as an Apple iMac computer. As shown, the computer system 1 of FIG. 1 includes a microprocessor 10, a read-only memory (ROM) 11, random access memory (RAM) 12, each connected to a bus system 18. The bus system 18 may include one or more buses connected to each other through various bridges, controllers and/or adapters, such as are well-known in the art. For example, the bus system may include a "system bus" that is connected through an adapter to one or more

> expansion buses, such as a Peripheral Component Interconnect (PCI) bus, or the like. Also coupled to the bus system 18 are a mass storage device 13, a display device 14, a keyboard 15, a pointing device 16, a communication device 17, and non-volatile RAM (NVRAM) 20. A cache memory 19 is coupled to the microprocessor 10.

Esfahani, '695, 3:1-22; Esfahani '265, 3:4-26

> The Boot Info file 40 resides on the boot device (e.g., a disk, or on a network) and has a localizable name. Identification information that leads to the file's path may be stored in NVRAM 20 or another suitable location, and the search algorithm for a usable Boot Info file parallels the search mechanism across SCSI, ATA, etc., used in the earlier Macintosh OS's StartSearch routine. By default, the Boot Info file 40 is located by using the current, active System folder's dirID (directory ID) in the boot block of each partition of the Hierarchical File System (HFS) and then searching for a file with a predetermined file type. Searching by file type is done to allow localization of the file.

Esfahani, '695, 7:66-8:10; Esfahani '265, 8:3-14

See also Esfahani Fig. 1

108. The method of claim 2, further comprising: compressing at least a portion of the additional boot data.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 2, further comprising: compressing at least a portion of the additional boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 2, further comprising: compressing at least a portion of the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 1.1, 5, 9.1 and 8 above.	

109. The method of claim 108, further comprising: storing the compressed additional boot data.	Esfahani, as evidenced by the example citations below, discloses "the method of claim 108, further comprising: storing the compressed additional boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 108, further comprising: storing the compressed additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Esfahani discloses this limitation: *See* Claims 5.1, 8.1 and 9.1 above.	

Appendix C2
Invalidity of U.S. Patent 8,880,862 based on Lillich

U.S. Patent Nos. 5,619,698 to Lillich ("Lillich '698) and 5,790,856 to Lillich ("Lillich '856"), alone or in combination, invalidate claims 1-6, 8-11, 13-17, 19, 23-24, 27-29, 31-33, 35-36, 39-40, 43-45, 47-53, 59-60, 63-65, 67, 71-72, 75-77, 79-81, 83-84, 87-89, 91-93, 97-98, and 107-109 of United States Patent No. 8,880,862 ("the '862 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

In addition, the prior art 68k operating system disclosed in Lillich (see Lillich '856, 1:20-5:55, and Lillich '698, 1:10-5:30) ("68k System") invalidates claims 1-6, 8-11, 13-17, 19, 23-24, 27-29, 31-33, 35-36, 39-40, 43-45, 47-53, 59-60, 63-65, 67, 71-72, 75-77, 79-81, 83-84, 87-89, 91-93, 97-98, and 107-10of the '862 Patent pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 because the system was on sale or in public use more than one year prior to the filing date of the '862 patent. Lillich '698, Lillich '856, and the 68k System are collectively referred to herein as "Lillich."

Additionally, Apple intends to provide and seek discovery related to this system to obtain copies of relevant materials, including but not limited to, architectural documents, design documents, implementation documents, internal publications, patent applications and business records relating to this product and will supplement these contentions after those materials are discovered or received. Apple also reserves its rights to rely on any additional 68k System materials, either individually or collectively, as prior art publications to the '862 patent.

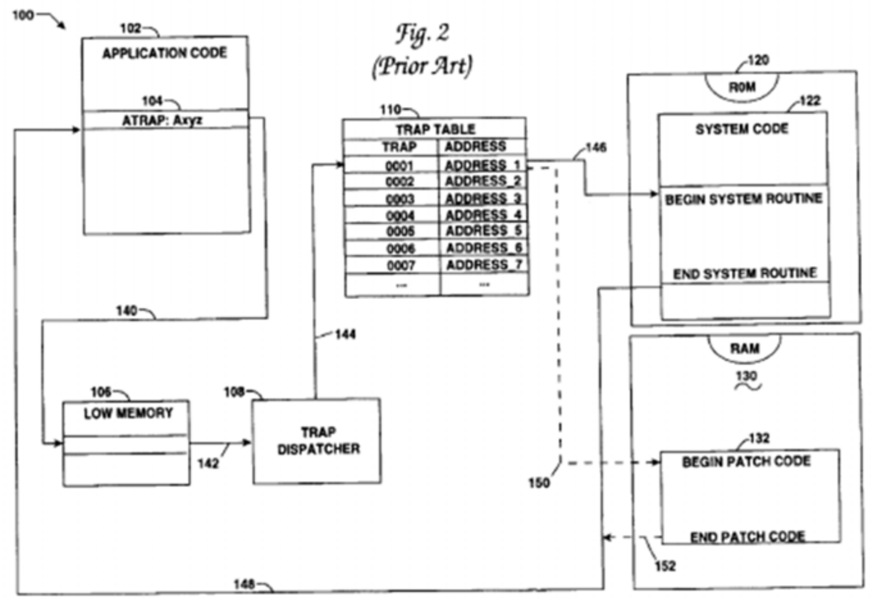
The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '862 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix C2
Invalidity of U.S. Patent 8,880,862 based on Lillich

1 (Preamble) A method for providing accelerated loading of an operating system in a computer system, the method comprising:	Lillich, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, the method comprising:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system in a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lillich discloses this claim limitation:

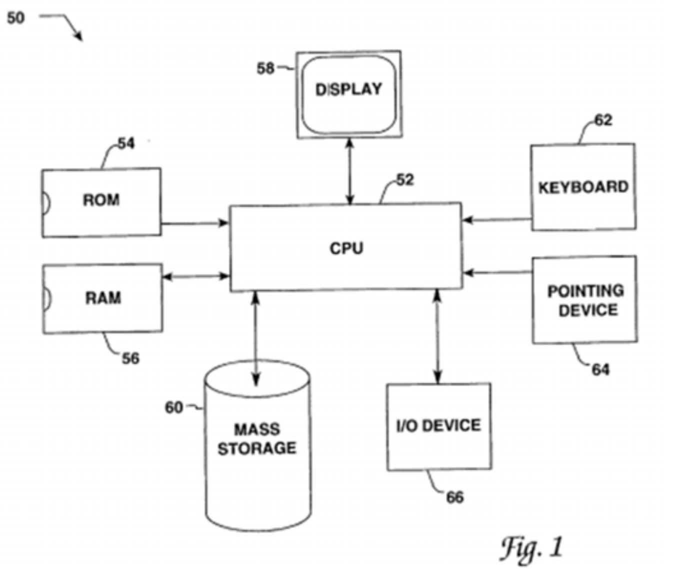


Lillich, Fig. 2

> "With reference to FIG. 2. the well known 68K patching paradigm 100 will be described. By way of background. The 68K patching paradigm 100 is implemented by versions of the Macintosh® Operating system designed for the Motorola 68K series of microprocessors. Which operating system is hereinafter referred to as the "68K operating system." The paradigm 100 is implemented on a computer system such as computer system 50 of FIG. 1. wherein a CPU 52 includes one of the Motorola 68K series microprocessors or microcontrollers as well as

other components necessary to properly interface with and control other devices coupled with the CPU 52."

Lillich '856, 1:40-51; *also* Lillich '698, 2:1-12.



Lillich, Fig. 1.

"By way of example, a representative computer system 50 is illustrated schematically in FIG. 1."

Lillich '856, 8:44-46; also Lillich '698, 9:39-41.

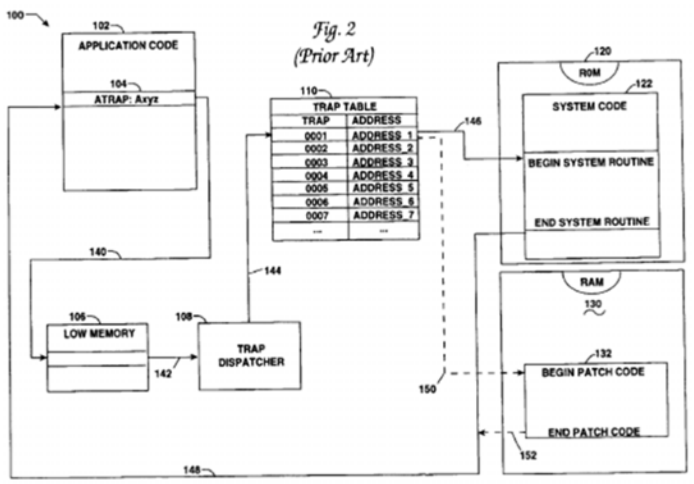
"As will be appreciated by those skilled in the art. CPU 52 includes a microprocessor and any additional circuitry and/ or device drivers necessary to control the computer system. For instance. the CPU 52 may include a keyboard controller which provides an interface between the microprocessor and the keyboard 62. ROM 64 is typically persistent memory accessible by the CPU 52 which contains the operating system instructions either in an executable format or in a compressed format which is expanded when the computer system 50 boots. RAM 56 is typically transient memory and is used as "scratch pad" memory by the operating system and/or any applications implemented on the computer system 50. For example. if a portion of the operating system present in ROM 64 is in compressed format, it may be expanded and stored into RAM 56."

Lillich '856, 8:53-67; *also* Lillich '698, 9:49-63.

1.1 loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory;	Lillich, as evidenced by the example citations below, discloses "loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lillich discloses this claim limitation:



Lillich, Fig. 2

> "With reference to FIG. 2. the well known 68K patching paradigm 100 will be described. By way of background. The 68K patching paradigm 100 is implemented by versions of the Macintosh® Operating system designed for the Motorola 68K series of microprocessors. Which operating system is hereinafter referred to as the "68K operating system." The paradigm 100 is implemented on a computer system such as computer system 50 of FIG. 1. wherein a CPU 52 includes one of the Motorola 68K series microprocessors or microcontrollers as well as

> other components necessary to properly interface with and control other devices coupled with the CPU 52."

Lillich '856, 1:40-51; *also* Lillich '698, 2:1-12.

> "The system routines for the 68K operating system reside mainly in ROM. However. to provide flexibility for any subsequent development. application code written for execution within the 68K operating system must be kept free of any specific ROM addresses. For this reason, all calls to system routines are passed indirectly through a trap table resident in RAM.

Lillich '856, 1:52-58, *also* Lillich '698, 2:14:20.

> "While the operating system routines reside mainly in ROM 120 (in their original state) information regarding the locations of the operating system routines is encoded in compressed form within ROM 120. Upon system start up. this information is decompressed and the trap table 110 is formed in RAM 130."

Lillich '856, 1:66-2:4, *also* Lillich, 2:28-33.

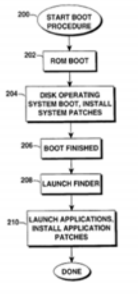

200
START BOOT PROCEDURE
202
ROM BOOT
204
DISK OPERATING SYSTEM BOOT, INSTALL SYSTEM PATCHES
206
BOOT FINISHED
208
LAUNCH FINDER
210
LAUNCH APPLICATIONS, INSTALL APPLICATION PATCHES
DONE

Fig. 3
(Prior Art)

Lillich '698, Fig 3.

> "Next, in reference to FIG. 3, one method for installing patches in the 68K operating system will be described. In an initial step 200, the computer boot process is started. Then, in a step 202, the ROM boot is performed. In ROM boot step 202, initialization procedures such as expanding the trap table from ROM into RAM are performed. Next, in a step 204, the disk operating system boot is performed. At this point any system patches are also installed."

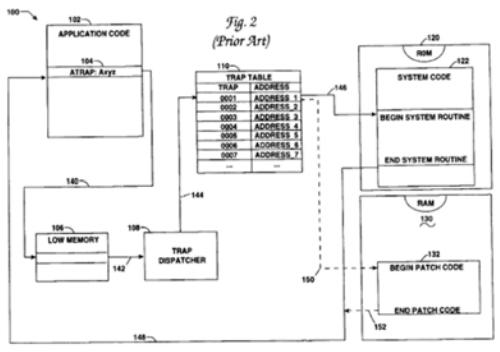
Lillich '698, 3:52-59.

Appendix C2
Invalidity of U.S. Patent 8,880,862 based on Lillich

1.2 accessing the loaded portion of the boot data in the compressed form from memory;	Lillich, as evidenced by the example citations below, discloses "accessing the loaded portion of the boot data in the compressed form from memory;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded portion of the boot data in the compressed form from memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lillich discloses this claim limitation:



Lillich, Fig. 2

> "With reference to FIG. 2. the well known 68K patching paradigm 100 will be described. By way of background. The 68K patching paradigm 100 is implemented by versions of the Macintosh® Operating system designed for the Motorola 68K series of microprocessors. Which operating system is hereinafter referred to as the "68K operating system." The paradigm 100 is implemented on a computer system such as computer system 50 of FIG. 1. wherein a CPU 52 includes one of the Motorola 68K series microprocessors or microcontrollers as well as other components necessary to properly interface with and control other devices coupled with the CPU 52."

Lillich '856, 1:40-51; *also* Lillich '698, 2:1-12.

> "The system routines for the 68K operating system reside mainly in ROM. However. to provide flexibility for any subsequent development. application code written for execution within the 68K operating system must be kept free of any specific ROM addresses. For this reason, all

> calls to system routines are passed indirectly through a trap table resident in RAM.

Lillich ’856, 1:52-58, *also* Lillich ’698, 2:14:20.

> “While the operating system routines reside mainly in ROM 120 (in their original state) information regarding the locations of the operating system routines is encoded in compressed form within ROM 120. Upon system start up. this information is decompressed and the trap table 110 is formed in RAM 130.”

Lillich ’856, 1:66-2:4, *also* Lillich, 2:28-33.

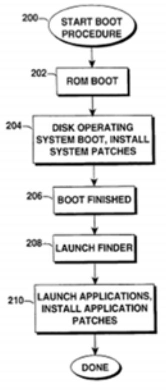

200
START BOOT PROCEDURE
202
ROM BOOT
204
DISK OPERATING SYSTEM BOOT, INSTALL SYSTEM PATCHES
206
BOOT FINISHED
208
LAUNCH FINDER
210
LAUNCH APPLICATIONS, INSTALL APPLICATION PATCHES
DONE

Fig. 3
(Prior Art)

Lillich ’698, Fig 3.

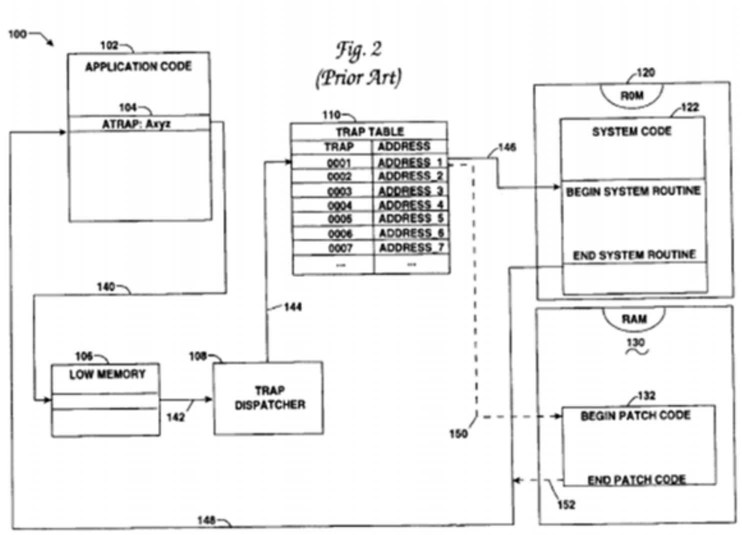
> “Next, in reference to FIG. 3, one method for installing patches in the 68K operating system will be described. In an initial step 200, the computer boot process is started. Then, in a step 202, the ROM boot is performed. In ROM boot step 202, initialization procedures such as expanding the trap table from ROM into RAM are performed. Next, in a step 204, the disk operating system boot is performed. At this point any system patches are also installed.”

Lillich ’698, 3:52-59.

1.3 decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form; and	Lillich, as evidenced by the example citations below, discloses "decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lillich discloses this claim limitation:



Lillich, Fig. 2

> "With reference to FIG. 2. the well known 68K patching paradigm 100 will be described. By way of background. The 68K patching paradigm 100 is implemented by versions of the Macintosh® Operating system designed for the Motorola 68K series of microprocessors. Which operating system is hereinafter referred to as the "68K operating system." The paradigm 100 is implemented on a computer system such as computer system 50 of FIG. 1. wherein a CPU 52 includes one of the Motorola 68K series microprocessors or microcontrollers as well as

> other components necessary to properly interface with and control other devices coupled with the CPU 52."

Lillich '856, 1:40-51; *also* Lillich '698, 2:1-12.

> "The system routines for the 68K operating system reside mainly in ROM. However. to provide flexibility for any subsequent development. application code written for execution within the 68K operating system must be kept free of any specific ROM addresses. For this reason, all calls to system routines are passed indirectly through a trap table resident in RAM.

Lillich '856, 1:52-58, *also* Lillich '698, 2:14:20.

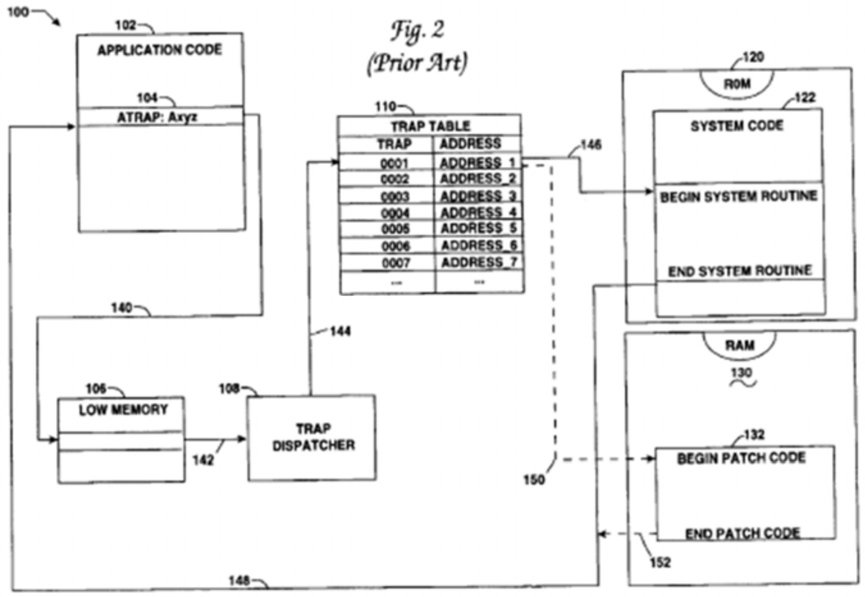
> "While the operating system routines reside mainly in ROM 120 (in their original state) information regarding the locations of the operating system routines is encoded in compressed form within ROM 120. Upon system start up. this information is decompressed and the trap table 110 is formed in RAM 130."

Lillich '856, 1:66-2:4, *also* Lillich, 2:28-33.

1.4.1 updating the boot data list,	Lillich, as evidenced by the example citations below, discloses "updating the boot data list,"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lillich discloses this claim limitation:



Lillich, Fig. 2

> "With reference to FIG. 2. the well known 68K patching paradigm 100 will be described. By way of background. The 68K patching paradigm 100 is implemented by versions of the Macintosh® Operating system designed for the Motorola 68K series of microprocessors. Which operating system is hereinafter referred to as the "68K operating system." The paradigm 100 is implemented on a computer system such as computer system 50 of FIG. 1. wherein a CPU 52 includes one of the Motorola 68K series microprocessors or microcontrollers as well as other components necessary to properly interface with and control other devices coupled with the CPU 52."

Lillich '856, 1:40-51; *also* Lillich '698, 2:1-12.

"The system routines for the 68K operating system reside mainly in ROM. However. to provide flexibility for any subsequent development. application code written for execution within the 68K operating system must be kept free of any specific ROM addresses. For this reason, all calls to system routines are passed indirectly through a trap table resident in RAM.

Lillich '856, 1:52-58, *also* Lillich '698, 2:14:20.

"While the operating system routines reside mainly in ROM 120 (in their original state) information regarding the locations of the operating system routines is encoded in compressed form within ROM 120. Upon system start up. this information is decompressed and the trap table 110 is formed in RAM 130."

Lillich '856, 1:66-2:4, *also* Lillich, 2:28-33.

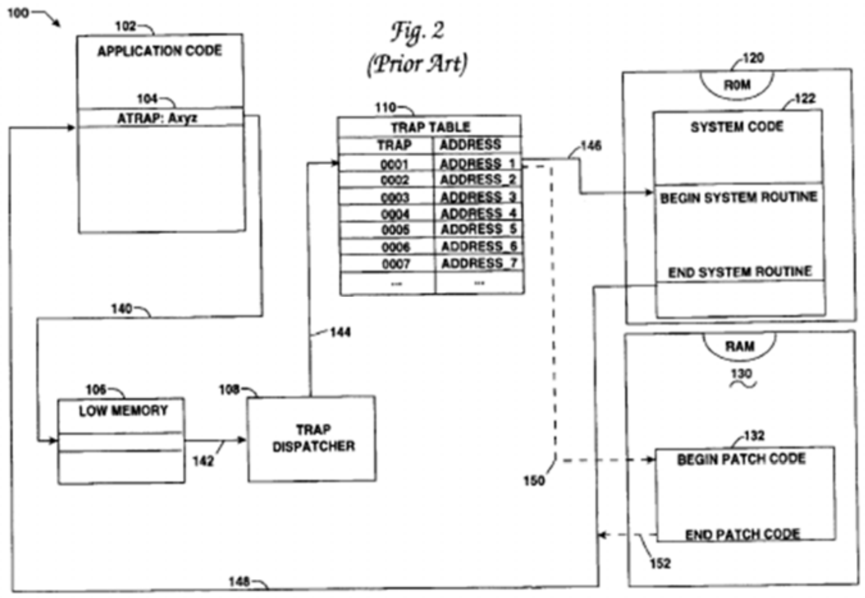
"Because the trap table 110 is resident in RAM 130. individual entries in the trap table 110 can be changed to point to addresses other than the original ROM addresses. This allows the system routines to be replaced or augmented by patches. At startup time the system can load new versions of individual routines (e.g. from the System file or from a floppy disk) into RAM and then patch the trap table in order to redirect any calls to the original system routine to the new versions."

Lillich '856, 2:52-60, *also* Lillich '698, 3:26-34.

1.4.2 wherein the decompressed portion of boot data comprises a portion of the operating system.	Lillich, as evidenced by the example citations below, discloses "wherein the decompressed portion of boot data comprises a portion of the operating system."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the decompressed portion of boot data comprises a portion of the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lillich discloses this claim limitation:



Lillich, Fig. 2

> "With reference to FIG. 2. the well known 68K patching paradigm 100 will be described. By way of background. The 68K patching paradigm 100 is implemented by versions of the Macintosh® Operating system designed for the Motorola 68K series of microprocessors. Which operating system is hereinafter referred to as the "68K operating system." The paradigm 100 is implemented on a computer system such as computer system 50 of FIG. 1. wherein a CPU 52 includes one of the Motorola 68K series microprocessors or microcontrollers as well as

other components necessary to properly interface with and control other devices coupled with the CPU 52."

Lillich '856, 1:40-51; *also* Lillich '698, 2:1-12.

"The system routines for the 68K operating system reside mainly in ROM. However. to provide flexibility for any subsequent development. application code written for execution within the 68K operating system must be kept free of any specific ROM addresses. For this reason, all calls to system routines are passed indirectly through a trap table resident in RAM.

Lillich '856, 1:52-58, *also* Lillich '698, 2:14:20.

"While the operating system routines reside mainly in ROM 120 (in their original state) information regarding the locations of the operating system routines is encoded in compressed form within ROM 120. Upon system start up. this information is decompressed and the trap table 110 is formed in RAM 130."

Lillich '856, 1:66-2:4, *also* Lillich, 2:28-33.

2. The method of claim 1, wherein the updating comprises: associating additional boot data with the boot data list.	Lillich, as evidenced by the example citations below, discloses "wherein the updating comprises: associating additional boot data with the boot data list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein associating additional boot data with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.4.1 above.	

3. The method of claim 1, wherein the updating comprises: removing an association of additional boot data that is associated with the boot data list from the boot data list.	Lillich, as evidenced by the example citations below, discloses "wherein the updating comprises: removing an association of additional boot data that is associated with the boot data list from the boot data list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein removing an association of additional boot data that is associated with the boot data list from the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.4.1 above.	

4. The method of claim 1, wherein the updating comprises: associating additional boot data with the boot data list; and compressing a portion of the additional boot data.	Lillich, as evidenced by the example citations below, discloses "wherein the updating comprises: associating additional boot data with the boot data list; and compressing a portion of the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein associating additional boot data with the boot data list; and compressing a portion of the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 1.4.1, and 2 above.	

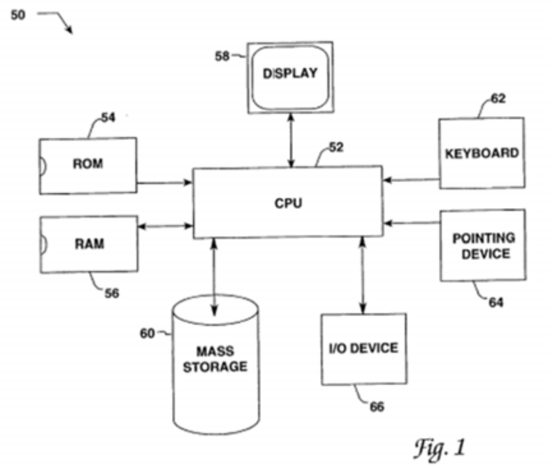
5 (Preamble) A method for booting a computer system, the method comprising:	Lillich, as evidenced by the example citations below, discloses "a method for booting a computer system, the method comprising:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lillich discloses this limitation:

See Claims 1-1.4.2 above.

See also



Lillich, Fig. 1.

> "By way of example, a representative computer system 50 is illustrated schematically in FIG. 1."

Lillich '856, 8:44-46; *also* Lillich '698, 9:39-41.

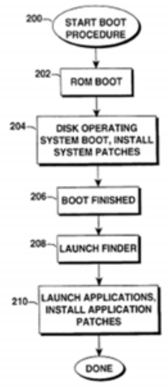

200
START BOOT PROCEDURE
202
ROM BOOT
204
DISK OPERATING SYSTEM BOOT, INSTALL SYSTEM PATCHES
206
BOOT FINISHED
208
LAUNCH FINDER
210
LAUNCH APPLICATIONS, INSTALL APPLICATION PATCHES
DONE

Fig. 3
(Prior Art)

Lillich '698, Fig 3.

> "Next, in reference to FIG. 3, one method for installing patches in the 68K operating system will be described. In an initial step 200, the computer boot process is started. Then, in a step 202, the ROM boot is performed. In ROM boot step 202, initialization procedures such as expanding the trap table from ROM into RAM are performed. Next, in a step 204, the disk operating system boot is performed. At this point any system patches are also installed."

Lillich '698, 3:52-59.

5.1 storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory;	Lillich, as evidenced by the example citations below, discloses "storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.1 above.	

5.2 loading the stored compressed boot data from the first memory;	Lillich, as evidenced by the example citations below, discloses "loading the stored compressed boot data from the first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the stored compressed boot data from the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.1 above.	

5.3 accessing the loaded compressed boot data;	Lillich, as evidenced by the example citations below, discloses "accessing the loaded compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.2 above.	

5.4 decompressing the accessed compressed boot data;	Lillich, as evidenced by the example citations below, discloses "decompressing the accessed compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.3 above.	

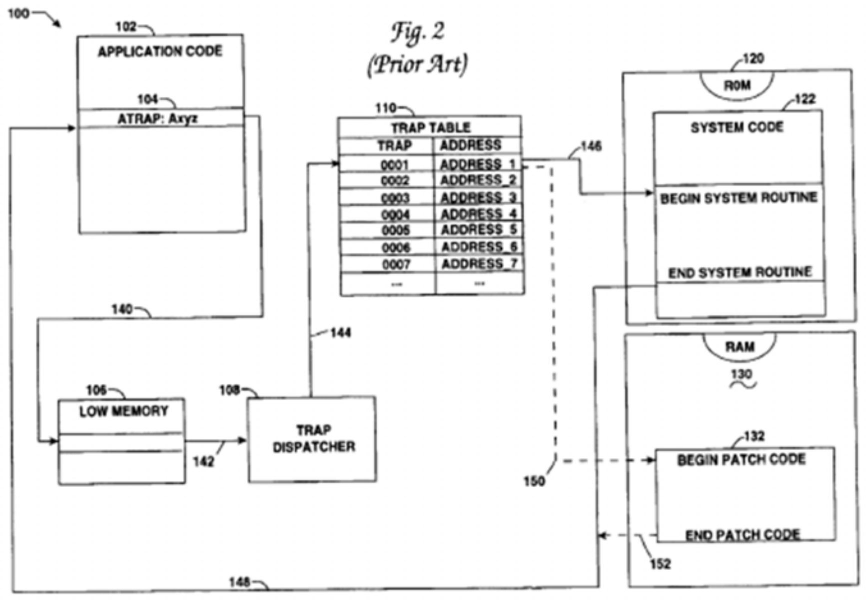
5.5 utilizing the decompressed boot data to at least partially boot the computer system;	Lillich, as evidenced by the example citations below, discloses “utilizing the decompressed boot data to at least partially boot the computer system”

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed boot data to at least partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.

Lillich discloses this limitation:

See Claims 1.3 and 1.4.2 above.

See also



Lillich, Fig. 2

> “With reference to FIG. 2. the well known 68K patching paradigm 100 will be described. By way of background. The 68K patching paradigm 100 is implemented by versions of the Macintosh® Operating system designed for the Motorola 68K series of microprocessors. Which operating system is hereinafter referred to as the “68K operating system.” The paradigm 100 is implemented on a computer system such

as computer system 50 of FIG. 1. wherein a CPU 52 includes one of the Motorola 68K series microprocessors or microcontrollers as well as other components necessary to properly interface with and control other devices coupled with the CPU 52."

Lillich '856, 1:40-51; *also* Lillich '698, 2:1-12.

"The system routines for the 68K operating system reside mainly in ROM. However. to provide flexibility for any subsequent development. application code written for execution within the 68K operating system must be kept free of any specific ROM addresses. For this reason, all calls to system routines are passed indirectly through a trap table resident in RAM.

Lillich '856, 1:52-58, *also* Lillich '698, 2:14:20.

"While the operating system routines reside mainly in ROM 120 (in their original state) information regarding the locations of the operating system routines is encoded in compressed form within ROM 120. Upon system start up. this information is decompressed and the trap table 110 is formed in RAM 130."

Lillich '856, 1:66-2:4, *also* Lillich, 2:28-33.

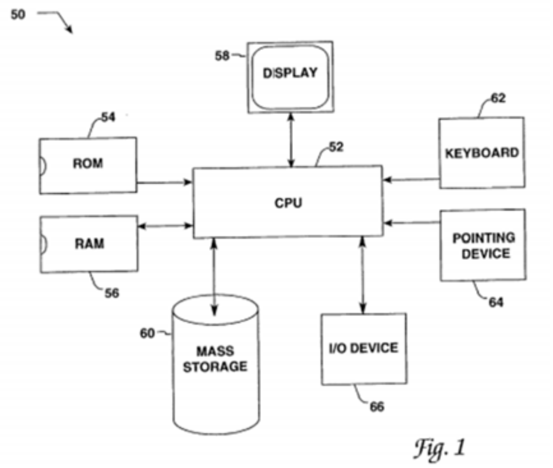
5.6 updating the boot data list;	Lillich, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.4.1 above.	

5.7 wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form.	Lillich, as evidenced by the example citations below, discloses "wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 1-1.3 above.	

6 (Preamble) A system comprising: a processor;	Lillich, as evidenced by the example citations below, discloses "a system comprising: a processor:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system comprising: a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lillich discloses this limitation:



Lillich, Fig. 1.

> "By way of example, a representative computer system 50 is illustrated schematically in FIG. 1."

Lillich '856, 8:44-46; *also* Lillich '698, 9:39-41.

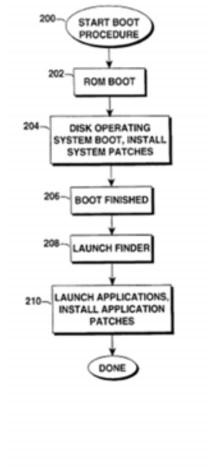

200
START BOOT PROCEDURE
202
ROM BOOT
204
DISK OPERATING SYSTEM BOOT, INSTALL SYSTEM PATCHES
206
BOOT FINISHED
208
LAUNCH FINDER
210
LAUNCH APPLICATIONS, INSTALL APPLICATION PATCHES
DONE

Fig. 3
(Prior Art)

Lillich '698, Fig 3.

> "Next, in reference to FIG. 3, one method for installing patches in the 68K operating system will be described. In an initial step 200, the computer boot process is started. Then, in a step 202, the ROM boot is performed. In ROM boot step 202, initialization procedures such as expanding the trap table from ROM into RAM are performed. Next, in a step 204, the disk operating system boot is performed. At this point any system patches are also installed."

Lillich '698, 3:52-59.

6.1 a memory; and	Lillich, as evidenced by the example citations below, discloses "a memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a memory.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 1.1 and 1.2 above.	

6.2 a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor	Lillich, as evidenced by the example citations below, discloses "a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 1.1, 1.2, and Claim 6 (Preamble) above.	

6.3 wherein the processor is configured:	Lillich, as evidenced by the example citations below, discloses "wherein the processor is configured:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the processor is configured), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 6 (Preamble) above	

6.4 to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory,	Lillich, as evidenced by the example citations below, discloses "to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.1 above.	

6.5 to access the loaded portion of the boot data in the compressed form,	Lillich, as evidenced by the example citations below, discloses "to access the loaded portion of the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to access the loaded portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.2 above.	

6.6 to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data, and	Lillich, as evidenced by the example citations below, discloses "to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.3 above.	

6.7 to update the boot data list.	Lillich, as evidenced by the example citations below, discloses "to update the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to update the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.4.1 above.	

Appendix C2
Invalidity of U.S. Patent 8,880,862 based on Lillich

8 (Preamble) A method of loading an operating system for booting a computer system, comprising:	Lillich, as evidenced by the example citations below, discloses "A method of loading an operating system for booting a computer system, comprising:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method of loading an operating system for booting a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1 (Preamble) above.	

8.1 storing a portion of the operating system in a compressed form in a first memory;	Lillich, as evidenced by the example citations below, discloses "storing a portion of the operating system in a compressed form in a first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing a portion of the operating system in a compressed form in a first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 6.2 above.	

8.2 loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list;	Lillich, as evidenced by the example citations below, discloses "loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.1 above.	

8.3 accessing the loaded portion of the operating system from the second memory in the compressed form;	Lillich, as evidenced by the example citations below, discloses "accessing the loaded portion of the operating system from the second memory in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded portion of the operating system from the second memory in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.2 above.	

8.4 decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system;	Lillich, as evidenced by the example citations below, discloses "decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 1.3 and 1.4.2 above.	

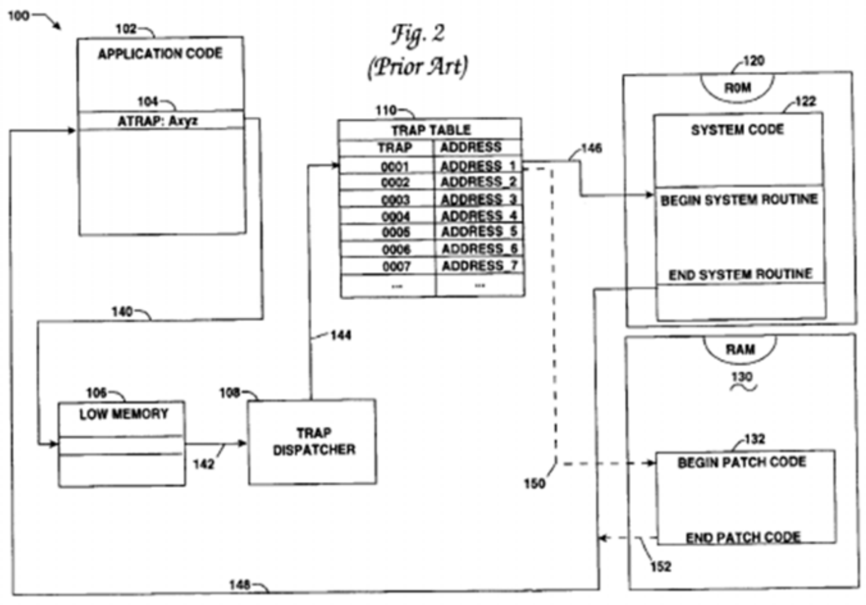
8.5 utilizing the decompressed portion of the operating system to at least partially boot the computer system; and	Lillich, as evidenced by the example citations below, discloses "utilizing the decompressed portion of the operating system to at least partially boot the computer system"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed portion of the operating system to at least partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lillich discloses this limitation:

See Claims 1.3 and 1.4.2 above.

See also



Lillich, Fig. 2

> "With reference to FIG. 2. the well known 68K patching paradigm 100 will be described. By way of background. The 68K patching paradigm 100 is implemented by versions of the Macintosh® Operating system designed for the Motorola 68K series of microprocessors. Which

> operating system is hereinafter referred to as the "68K operating system." The paradigm 100 is implemented on a computer system such as computer system 50 of FIG. 1. wherein a CPU 52 includes one of the Motorola 68K series microprocessors or microcontrollers as well as other components necessary to properly interface with and control other devices coupled with the CPU 52."

Lillich '856, 1:40-51; *also* Lillich '698, 2:1-12.

> "The system routines for the 68K operating system reside mainly in ROM. However. to provide flexibility for any subsequent development. application code written for execution within the 68K operating system must be kept free of any specific ROM addresses. For this reason, all calls to system routines are passed indirectly through a trap table resident in RAM.

Lillich '856, 1:52-58, *also* Lillich '698, 2:14:20.

> "While the operating system routines reside mainly in ROM 120 (in their original state) information regarding the locations of the operating system routines is encoded in compressed form within ROM 120. Upon system start up. this information is decompressed and the trap table 110 is formed in RAM 130."

Lillich '856, 1:66-2:4, *also* Lillich, 2:28-33.

8.6 updating the boot data list,	Lillich, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.4.1 above.	

8.7 wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form.	Lillich, as evidenced by the example citations below, discloses "wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 1-1.3 and 5.7 above.	

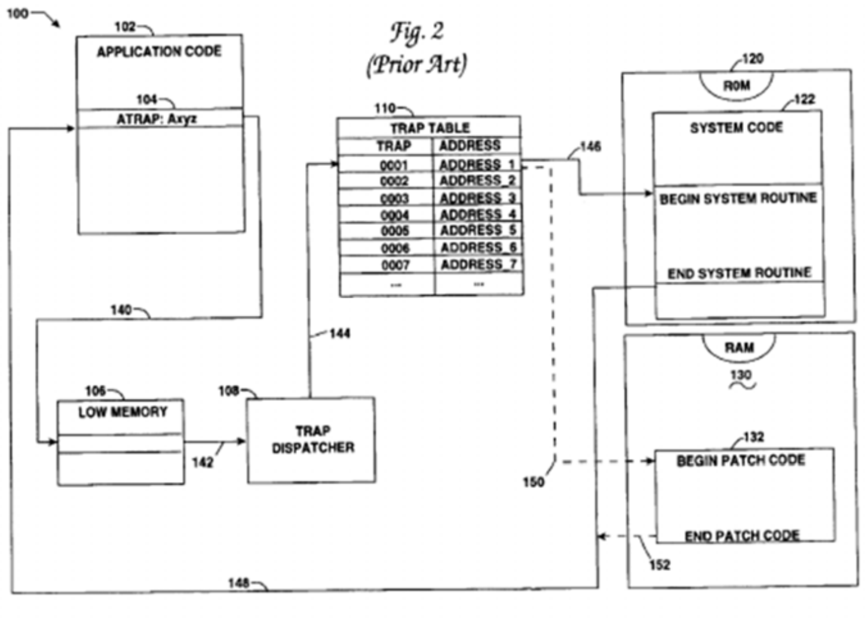
9.1 The method of claim 8, further comprising: compressing an additional portion of the operating system that is not associated with the boot data list; and	Lillich, as evidenced by the example citations below, discloses "the method of claim 8, further comprising: compressing an additional portion of the operating system that is not associated with the boot data list"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, compressing an additional portion of the operating system that is not associated with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lillich discloses this limitation:

See Claim 1.1 above.

See also



Lillich, Fig. 2

> "With reference to FIG. 2. the well known 68K patching paradigm 100 will be described. By way of background. The 68K patching paradigm

> 100 is implemented by versions of the Macintosh® Operating system designed for the Motorola 68K series of microprocessors. Which operating system is hereinafter referred to as the "68K operating system." The paradigm 100 is implemented on a computer system such as computer system 50 of FIG. 1. wherein a CPU 52 includes one of the Motorola 68K series microprocessors or microcontrollers as well as other components necessary to properly interface with and control other devices coupled with the CPU 52."

Lillich '856, 1:40-51; *also* Lillich '698, 2:1-12.

> "The system routines for the 68K operating system reside mainly in ROM. However. to provide flexibility for any subsequent development. application code written for execution within the 68K operating system must be kept free of any specific ROM addresses. For this reason, all calls to system routines are passed indirectly through a trap table resident in RAM.

Lillich '856, 1:52-58, *also* Lillich '698, 2:14:20.

> "While the operating system routines reside mainly in ROM 120 (in their original state) information regarding the locations of the operating system routines is encoded in compressed form within ROM 120. Upon system start up. this information is decompressed and the trap table 110 is formed in RAM 130."

Lillich '856, 1:66-2:4, *also* Lillich, 2:28-33.

9.2 storing the additional portion of the operating system in the first memory, and	Lillich, as evidenced by the example citations below, discloses "storing the additional portion of the operating system in the first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing the additional portion of the operating system in the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 8.1 above.	

9.3 wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system.	Lillich, as evidenced by the example citations below, discloses "wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 8.5 above.	

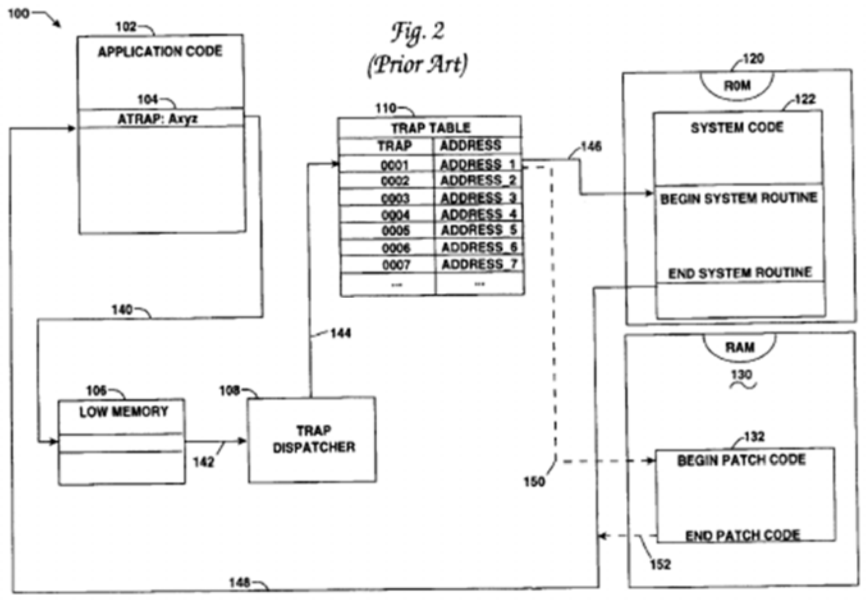
10. The method of claim 9, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder.	Lillich, as evidenced by the example citations below, discloses "the method of claim 9, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lillich discloses this limitation:

See Claim 9.1 above.

See also



Lillich, Fig. 2

> "With reference to FIG. 2. the well known 68K patching paradigm 100 will be described. By way of background. The 68K patching paradigm

100 is implemented by versions of the Macintosh® Operating system designed for the Motorola 68K series of microprocessors. Which operating system is hereinafter referred to as the "68K operating system." The paradigm 100 is implemented on a computer system such as computer system 50 of FIG. 1. wherein a CPU 52 includes one of the Motorola 68K series microprocessors or microcontrollers as well as other components necessary to properly interface with and control other devices coupled with the CPU 52."

Lillich '856, 1:40-51; *also* Lillich '698, 2:1-12.

"The system routines for the 68K operating system reside mainly in ROM. However. to provide flexibility for any subsequent development. application code written for execution within the 68K operating system must be kept free of any specific ROM addresses. For this reason, all calls to system routines are passed indirectly through a trap table resident in RAM.

Lillich '856, 1:52-58, *also* Lillich '698, 2:14:20.

"While the operating system routines reside mainly in ROM 120 (in their original state) information regarding the locations of the operating system routines is encoded in compressed form within ROM 120. Upon system start up. this information is decompressed and the trap table 110 is formed in RAM 130."

Lillich '856, 1:66-2:4, *also* Lillich, 2:28-33.

11 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	Lillich, as evidenced by the example citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, comprising:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1 (Preamble) above.	

11.1 loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system;	Lillich, as evidenced by the example citations below, discloses "loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.1 and 6 (Preamble) above.	

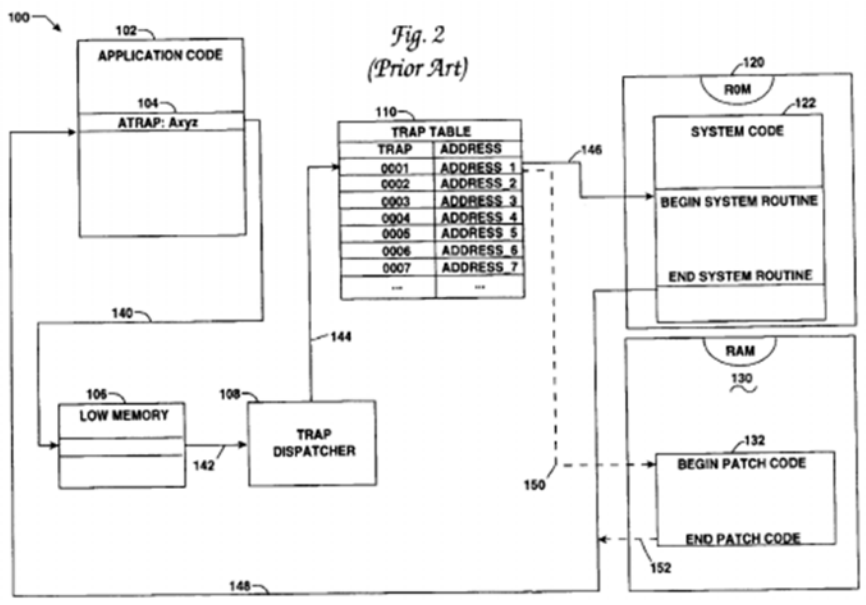
11.2 accessing the loaded boot data in compressed form from the memory;	Lillich, as evidenced by the example citations below, discloses "accessing the loaded boot data in compressed form from the memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded boot data in compressed form from the memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.2 above.	

11.3 decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form;	Lillich, as evidenced by the example citations below, discloses "decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.3 above.	

11.4 utilizing the decompressed boot data to load at least a portion of the operating system for the computer system; and	Lillich, as evidenced by the example citations below, discloses "utilizing the decompressed boot data to load at least a portion of the operating system for the computer system"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed boot data to load at least a portion of the operating system for the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lillich discloses this limitation:



Lillich, Fig. 2

> "With reference to FIG. 2. the well known 68K patching paradigm 100 will be described. By way of background. The 68K patching paradigm 100 is implemented by versions of the Macintosh® Operating system designed for the Motorola 68K series of microprocessors. Which operating system is hereinafter referred to as the "68K operating system." The paradigm 100 is implemented on a computer system such as computer system 50 of FIG. 1. wherein a CPU 52 includes one of the Motorola 68K series microprocessors or microcontrollers as well as

> other components necessary to properly interface with and control other devices coupled with the CPU 52."

Lillich '856, 1:40-51; *also* Lillich '698, 2:1-12.

> "The system routines for the 68K operating system reside mainly in ROM. However. to provide flexibility for any subsequent development. application code written for execution within the 68K operating system must be kept free of any specific ROM addresses. For this reason, all calls to system routines are passed indirectly through a trap table resident in RAM.

Lillich '856, 1:52-58, *also* Lillich '698, 2:14:20.

> "While the operating system routines reside mainly in ROM 120 (in their original state) information regarding the locations of the operating system routines is encoded in compressed form within ROM 120. Upon system start up. this information is decompressed and the trap table 110 is formed in RAM 130."

Lillich '856, 1:66-2:4, *also* Lillich, 2:28-33.

11.5 updating the boot data list.	Lillich, as evidenced by the example citations below, discloses “updating the boot data list”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.4.1 above.	

13 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	Lillich, as evidenced by the example citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, comprising"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1 (Preamble) above.	

13.1 loading boot data in a compressed form that is associated with a boot data list from a boot device;	Lillich, as evidenced by the example citations below, discloses "loading boot data in a compressed form that is associated with a boot data list from a boot device"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading boot data in a compressed form that is associated with a boot data list from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.1 above.	

13.2 accessing the loaded boot data in the compressed form;	Lillich, as evidenced by the example citations below, discloses "accessing the loaded boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.2 above.	

13.3 decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form;	Lillich, as evidenced by the example citations below, discloses "decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.3 above.	

13.4 updating the boot data list.	Lillich, as evidenced by the example citations below, discloses “updating the boot data list”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.4.1 above.	

Appendix C2
Invalidity of U.S. Patent 8,880,862 based on Lillich

14 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	Lillich, as evidenced by the example citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, comprising"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1 (Preamble) above.	

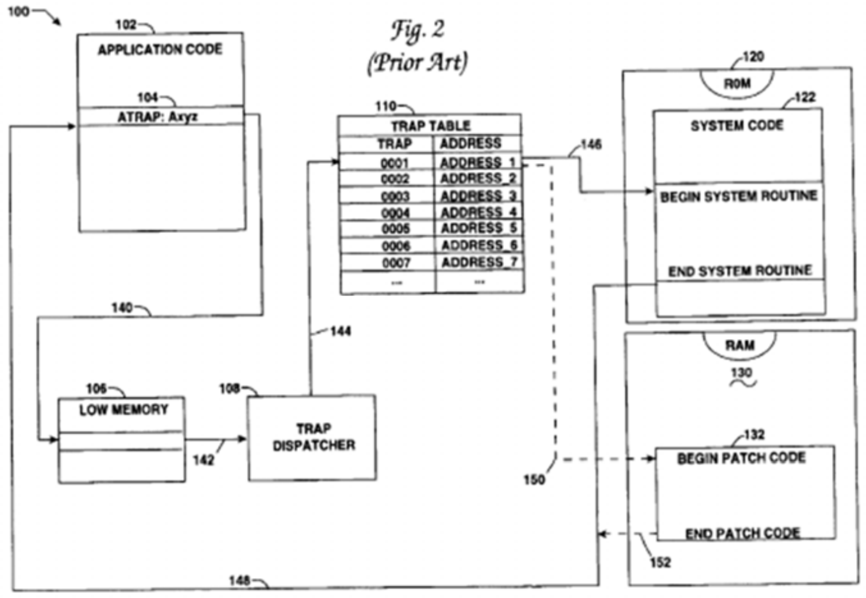
14.1 accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list;	Lillich, as evidenced by the example citations below, discloses "accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lillich discloses this limitation:

See Claims 1.1 and 1.2 above.

See also



Lillich, Fig. 2

> "With reference to FIG. 2. the well known 68K patching paradigm 100 will be described. By way of background. The 68K patching paradigm 100 is implemented by versions of the Macintosh® Operating system

designed for the Motorola 68K series of microprocessors. Which operating system is hereinafter referred to as the "68K operating system." The paradigm 100 is implemented on a computer system such as computer system 50 of FIG. 1. wherein a CPU 52 includes one of the Motorola 68K series microprocessors or microcontrollers as well as other components necessary to properly interface with and control other devices coupled with the CPU 52."

Lillich '856, 1:40-51; *also* Lillich '698, 2:1-12.

"The system routines for the 68K operating system reside mainly in ROM. However. to provide flexibility for any subsequent development. application code written for execution within the 68K operating system must be kept free of any specific ROM addresses. For this reason, all calls to system routines are passed indirectly through a trap table resident in RAM.

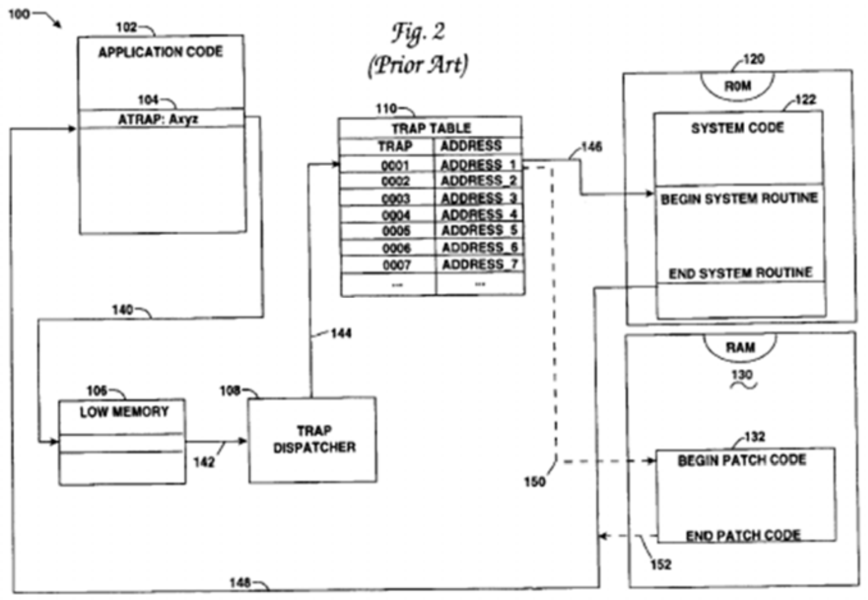
Lillich '856, 1:52-58, *also* Lillich '698, 2:14:20.

"While the operating system routines reside mainly in ROM 120 (in their original state) information regarding the locations of the operating system routines is encoded in compressed form within ROM 120. Upon system start up. this information is decompressed and the trap table 110 is formed in RAM 130."

Lillich '856, 1:66-2:4, *also* Lillich, 2:28-33.

14.2 loading the boot data into a memory; and	Lillich, as evidenced by the example citations below, discloses "loading the boot data into a memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the boot data into a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.1 above.	

14.3 servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form; and	Lillich, as evidenced by the example citations below, discloses "servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 1.2 and 1.3 above. *See also*	



Lillich, Fig. 2

> "With reference to FIG. 2. the well known 68K patching paradigm 100 will be described. By way of background. The 68K patching paradigm 100 is implemented by versions of the Macintosh® Operating system designed for the Motorola 68K series of microprocessors. Which operating system is hereinafter referred to as the "68K operating system." The paradigm 100 is implemented on a computer system such as computer system 50 of FIG. 1. wherein a CPU 52 includes one of the Motorola 68K series microprocessors or microcontrollers as well as other components necessary to properly interface with and control other devices coupled with the CPU 52."

Lillich '856, 1:40-51; *also* Lillich '698, 2:1-12.

> "The system routines for the 68K operating system reside mainly in ROM. However. to provide flexibility for any subsequent development. application code written for execution within the 68K operating system must be kept free of any specific ROM addresses. For this reason, all calls to system routines are passed indirectly through a trap table resident in RAM.

Lillich '856, 1:52-58, *also* Lillich '698, 2:14:20.

> "While the operating system routines reside mainly in ROM 120 (in their original state) information regarding the locations of the operating system routines is encoded in compressed form within ROM 120. Upon system start up. this information is decompressed and the trap table 110 is formed in RAM 130."

Lillich '856, 1:66-2:4, *also* Lillich, 2:28-33.

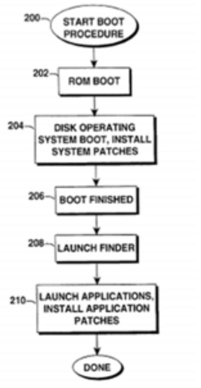

200
START BOOT PROCEDURE
202
ROM BOOT
204
DISK OPERATING SYSTEM BOOT, INSTALL SYSTEM PATCHES
206
BOOT FINISHED
208
LAUNCH FINDER
210
LAUNCH APPLICATIONS, INSTALL APPLICATION PATCHES
DONE

Fig. 3
(Prior Art)

Lillich '698, Fig 3.

> "Next, in reference to FIG. 3, one method for installing patches in the 68K operating system will be described. In an initial step 200, the computer boot process is started. Then, in a step 202, the ROM boot is performed. In ROM boot step 202, initialization procedures such as expanding the trap table from ROM into RAM are performed. Next, in a step 204, the disk operating system boot is performed. At this point any system patches are also installed."

Lillich '698, 3:52-59.

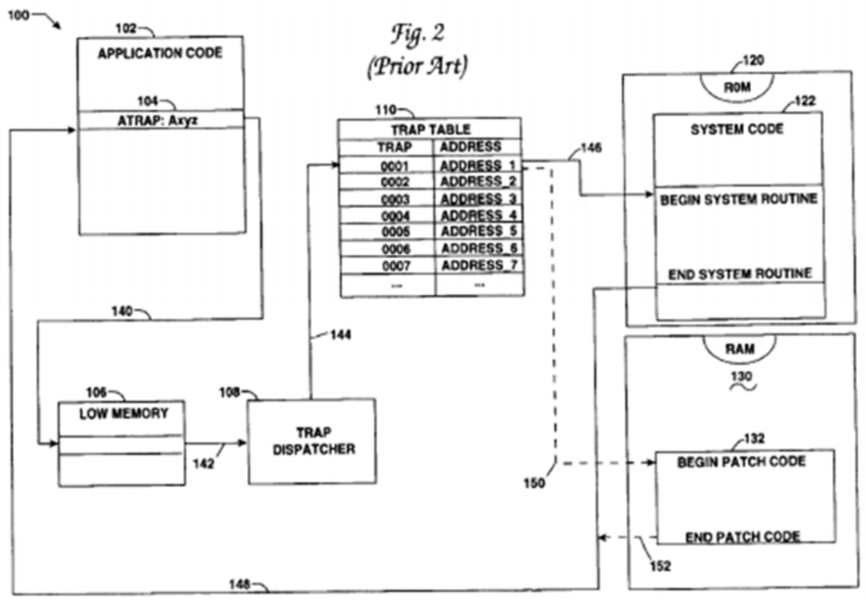
14.4 updating the boot data list.	Lillich, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.4.1 above.	

15. The method of claim 14, wherein the boot data comprises: a program code associated with the operating system.	Lillich, as evidenced by the example citations below, discloses "the method of claim 14, wherein the boot data comprises: a program code associated with the operating system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the boot data comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 1 (Preamble) and 1.1 above.	

16. The method of claim 14, wherein the operating system comprises: a plurality of files.	Lillich, as evidenced by the example citations below, discloses "the method of claim 14, wherein the operating system comprises: a plurality of files."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lillich discloses this limitation:



Lillich, Fig. 2

> "With reference to FIG. 2. the well known 68K patching paradigm 100 will be described. By way of background. The 68K patching paradigm 100 is implemented by versions of the Macintosh® Operating system designed for the Motorola 68K series of microprocessors. Which operating system is hereinafter referred to as the "68K operating system." The paradigm 100 is implemented on a computer system such as computer system 50 of FIG. 1. wherein a CPU 52 includes one of the Motorola 68K series microprocessors or microcontrollers as well as

other components necessary to properly interface with and control other devices coupled with the CPU 52."

Lillich '856, 1:40-51; *also* Lillich '698, 2:1-12.

"The system routines for the 68K operating system reside mainly in ROM. However. to provide flexibility for any subsequent development. application code written for execution within the 68K operating system must be kept free of any specific ROM addresses. For this reason, all calls to system routines are passed indirectly through a trap table resident in RAM.

Lillich '856, 1:52-58, *also* Lillich '698, 2:14:20.

"While the operating system routines reside mainly in ROM 120 (in their original state) information regarding the locations of the operating system routines is encoded in compressed form within ROM 120. Upon system start up. this information is decompressed and the trap table 110 is formed in RAM 130."

Lillich '856, 1:66-2:4, *also* Lillich, 2:28-33.

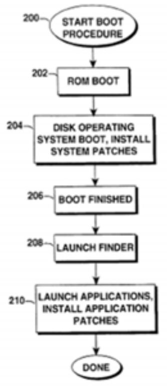

200
START BOOT PROCEDURE
202
ROM BOOT
204
DISK OPERATING SYSTEM BOOT, INSTALL SYSTEM PATCHES
206
BOOT FINISHED
208
LAUNCH FINDER
210
LAUNCH APPLICATIONS, INSTALL APPLICATION PATCHES
DONE

Fig. 3
(Prior Art)

Lillich '698, Fig 3.

"Next, in reference to FIG. 3, one method for installing patches in the 68K operating system will be described. In an initial step 200, the computer boot process is started. Then, in a step 202, the ROM boot is performed. In ROM boot step 202, initialization procedures such as expanding the trap table from ROM into RAM are performed. Next, in a step 204, the disk operating system boot is performed. At this point any

system patches are also installed."   Lillich '698, 3:52-59.

17. The method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program.	Lillich, as evidenced by the example citations below, discloses "the method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lillich discloses this limitation:

See Claim 15 above.

See also

> The system routines for the 68K operating system reside mainly in ROM. However, to provide flexibility for any subsequent development, application code written for execution within the 68K operating system must be kept free of any specific ROM addresses. For this reason, all calls to system routines are passed indirectly through a trap table resident in RAM. This indirect mechanism permits the ROM addressing of system routines to vary, or to be replaced by patch routines, without affecting the operation of applications which utilize the system routines.

Lillich '856, 1:52-61; *also* Lillich '698, 2:14-23

> Additionally, when new applications are launched they too can load new versions of individual routines into RAM and then patch the trap table in order to redirect any calls to the original system routine to the new versions.

Lillich '856, 2:60-64; *also* Lillich '698, 3:34-37

> In the world of Macintosh®, DLLs can be loosely divided into three different categories: applications, import libraries, and extensions. Typically, applications are fragments which have a user interface and are designed to function interactively with the user. Often applications do not export symbols to other fragments but rather serve as a root fragments, providing some root functionality and importing other necessary functionality.

> . . .
>
> Once an application has been launched and the binding manager has completed preparation, the end result is a new, self-sufficient executable process with an associated data closure. The closure contains a root fragment as well as instances of all the import libraries required to resolve all the import symbols present in both the root fragment and the import libraries. In explanation, the root fragment is the fragment which includes some root functionality as well as a list of import symbols.

Lillich ’856, 3:40-4:9

> As discussed previously, new versions of individual system routines may be loaded into RAM and the trap table patched in order to redirect any calls to the original system routine to the new versions. After this is complete, the operating system boot procedure is complete in a step 206. In a next step 208, the application "Finder" is launched. As will be appreciated by those skilled in the art, the Finder is the primal application which performs critical tasks such as displaying the Macintosh® desktop and launching other applications at the request of the user. Once the Finder is running, in a step 210 other applications may be launched and in turn any patches necessary for the other applications may be installed.

Lillich ’698, 3:60-4:5

19.1 The method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:	Lillich, as evidenced by the example citations below, discloses "the method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 1.1 and 1.2 above	

19.2 associating the accessed boot data that is not associated with the boot data list to the boot data list.	Lillich, as evidenced by the example citations below, discloses "associating the accessed boot data that is not associated with the boot data list to the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, associating the accessed boot data that is not associated with the boot data list to the boot data list.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 2 above.	

23. The method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files.	Lillich, as evidenced by the example citations below, discloses "the method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 16 above.	

24. The method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system.	Lillich, as evidenced by the example citations below, discloses “the method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system.
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Lillich discloses this limitation: *See* Claim 15 above.	

27. The method of claim 1, wherein the memory comprises: a physical memory.	Lillich, as evidenced by the example citations below, discloses "the method of claim 1, wherein the memory comprises: a physical memory."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lillich discloses this limitation:

See Claims 1.1 and 1.2 above

See also

> Mass storage device 60 is coupled with CPU 52 and may be any mass storage device such as a hard disk system, a floptical disk system, a tape drive or the like. Mass storage device 60 generally includes code fragments such as applications, import libraries, and extensions which are not currently in use by the system. I/O device 66 is coupled to the CPU 52 and may be a network card, a printer port, modem, etc. Additionally there may be a multiplicity of I/O devices such as I/O device 66 coupled to the computer system 50. Design and construction of computer systems such as computer system 50 will be well known to those skilled in the art.

Lillich '856, 9:7-18; *also* Lillich '698, 10:3-14

28. The method of claim 1, wherein the operating system comprises: a plurality of files.	Lillich, as evidenced by the example citations below, discloses "the method of claim 1, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 16 above.	

29. The method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program.	Lillich, as evidenced by the example citations below, discloses "the method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 15 and 17 above.	

31. The method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access.	Lillich, as evidenced by the example citations below, discloses "the method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 1.1 and 1.2 above	

32. The method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form.	Lillich, as evidenced by the example citations below, discloses "the method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 1.1 above	

33. The method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form.	Lillich, as evidenced by the example citations below, discloses "the method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 1.1 and 32 above.	

35. The method of claim 5, wherein the compressed boot data represents a plurality of files.	Lillich, as evidenced by the example citations below, discloses "the method of claim 5, wherein the compressed boot data represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the compressed boot data represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 16 and 23 above.	

36. The method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system.	Lillich, as evidenced by the example citations below, discloses "the method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 15 and 17 above.	

39. The method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory.	Lillich, as evidenced by the example citations below, discloses “the method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Lillich discloses this limitation: *See* Claims 1.1 and 1.2 above	

40. The method of claim 36, wherein the operating system comprises: a plurality of files.	Lillich, as evidenced by the example citations below, discloses “the method of claim 36, wherein the operating system comprises: a plurality of files”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 36, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Lillich discloses this limitation: *See* Claims 16 and 23 above.	

43. The method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access.	Lillich, as evidenced by the example citations below, discloses "the method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 31 above.	

44. The method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data.	Lillich, as evidenced by the example citations below, discloses "the method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 32 above.	

45. The method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data.	Lillich, as evidenced by the example citations below, discloses "the method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 32 and 33 above.	

47. The system of claim 6, wherein the boot data in the compressed form represents a plurality of files.	Lillich, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 16 and 23 above.	

48. The system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system.	Lillich, as evidenced by the example citations below, discloses “the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Lillich discloses this limitation: *See* Claims 15, 17 and 24 above.	

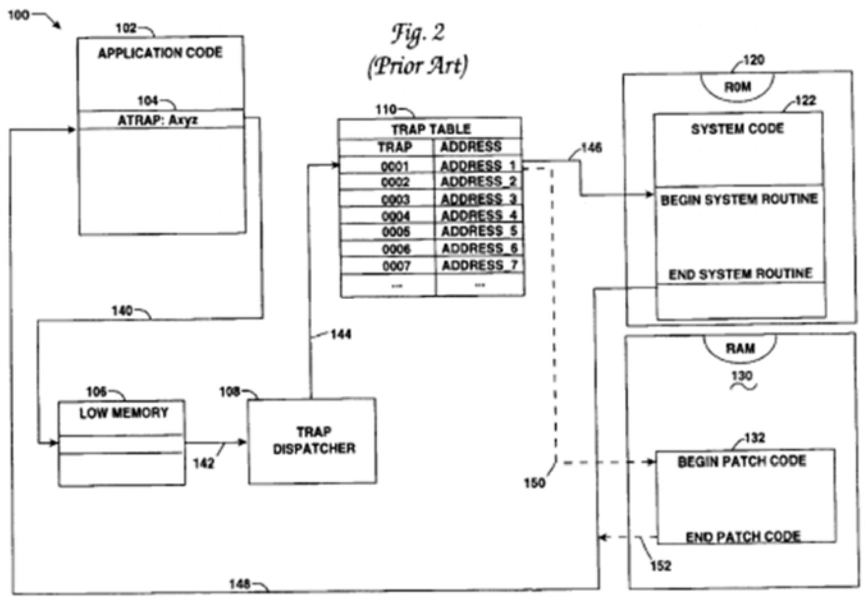
49. The system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form.	Lillich, as evidenced by the example citations below, discloses "the system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 10 above.	

Appendix C2
Invalidity of U.S. Patent 8,880,862 based on Lillich

50. The system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form.	Lillich, as evidenced by the example citations below, discloses "the system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lillich discloses this limitation:



Lillich, Fig. 2

> "With reference to FIG. 2. the well known 68K patching paradigm 100 will be described. By way of background. The 68K patching paradigm 100 is implemented by versions of the Macintosh® Operating system designed for the Motorola 68K series of microprocessors. Which operating system is hereinafter referred to as the "68K operating system." The paradigm 100 is implemented on a computer system such as computer system 50 of FIG. 1. wherein a CPU 52 includes one of the

> Motorola 68K series microprocessors or microcontrollers as well as other components necessary to properly interface with and control other devices coupled with the CPU 52."

Lillich '856, 1:40-51; *also* Lillich '698, 2:1-12.

> "The system routines for the 68K operating system reside mainly in ROM. However. to provide flexibility for any subsequent development. application code written for execution within the 68K operating system must be kept free of any specific ROM addresses. For this reason, all calls to system routines are passed indirectly through a trap table resident in RAM.

Lillich '856, 1:52-58, *also* Lillich '698, 2:14:20.

> "While the operating system routines reside mainly in ROM 120 (in their original state) information regarding the locations of the operating system routines is encoded in compressed form within ROM 120. Upon system start up. this information is decompressed and the trap table 110 is formed in RAM 130."

Lillich '856, 1:66-2:4, *also* Lillich, 2:28-33.

51. The system of claim 6, wherein the first memory comprises: a physical memory.	Lillich, as evidenced by the example citations below, discloses "the system of claim 6, wherein the first memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the first memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 27 and 39 above.	

52. The system of claim 6, wherein the boot data the compressed form comprises: a plurality of files.	Lillich, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data the compressed form comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data the compressed form comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 16 and 23 above.	

53. The system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program.	Lillich, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 15, 17 and 24 above.	

59. The method of claim 8, wherein the operating system in the compressed form represents a plurality of files.	Lillich, as evidenced by the example citations below, discloses “the method of claim 8, wherein the operating system in the compressed form represents a plurality of files”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Lillich discloses this limitation: *See* Claims 16 and 23 above.	

60. The method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system.	Lillich, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 15, 17 and 24 above.	

63. The method of claim 8, wherein the second memory comprises: a physical memory.	Lillich, as evidenced by the example citations below, discloses "the method of claim 8, wherein the second memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the second memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 27 and 38 above.	

64. The method of claim 8, wherein the operating system comprises: a plurality of files.	Lillich, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 16 and 23 above.	

65. The method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program.	Lillich, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 15, 17 and 24 above.	

67. The method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access.	Lillich, as evidenced by the example citations below, discloses "the method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 31 above.	

71. The method of claim 11, wherein the boot data in the compressed form represents a plurality of files.	Lillich, as evidenced by the example citations below, discloses "the method of claim 11, wherein the boot data in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 16 and 23 above.	

72. The method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system.	Lillich, as evidenced by the example citations below, discloses "the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 15, 17 and 24 above.	

75. The method of claim 11, wherein the memory comprises: a physical memory.	Lillich, as evidenced by the example citations below, discloses "the method of claim 11, wherein the memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 27 above.	

76. The method of claim 11, wherein the operating system comprises: a plurality of files.	Lillich, as evidenced by the example citations below, discloses "the method of claim 11, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 16 and 23 above.	

77. The method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program.	Lillich, as evidenced by the example citations below, discloses "the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 15, 17 and 24 above.	

79. The method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access.	Lillich, as evidenced by the example citations below, discloses "the method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 31 above.	

80. The method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form.	Lillich, as evidenced by the example citations below, discloses "the method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 32, 33 and 49 above.	

81. The method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form.	Lillich, as evidenced by the example citations below, discloses "the method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 32, 33 and 49 above.	

83. The method of claim 13, wherein the boot data in the compressed form represents a plurality of files.	Lillich, as evidenced by the example citations below, discloses "the method of claim 13, wherein the boot data in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 16 and 23 above.	

84. The method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system.	Lillich, as evidenced by the example citations below, discloses "the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 15, 17 and 24 above.	

87. The method of claim 13, wherein the memory comprises: a physical memory.	Lillich, as evidenced by the example citations below, discloses "the method of claim 13, wherein the memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 27 above.	

88. The method of claim 13, wherein the operating system comprises: a plurality of files.	Lillich, as evidenced by the example citations below, discloses "the method of claim 13, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 16 and 23 above.	

89. The method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program.	Lillich, as evidenced by the example citations below, discloses "the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 15, 17 and 24 above.	

91. The method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access.	Lillich, as evidenced by the example citations below, discloses "the method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 31 above.	

92. The method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data.	Lillich, as evidenced by the example citations below, discloses "the method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 32, 33 and 49 above.	

93. The method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data.	Lillich, as evidenced by the example citations below, discloses "the method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claim 32, 33, and 49 above.	

97.1 The method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list,	Lillich, as evidenced by the example citations below, discloses "the method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 9.1-9.3 and 19 above.	

97.2 and wherein the updating comprises: associating the additional compressed boot data with the boot data list.	Lillich, as evidenced by the example citations below, discloses “and wherein the updating comprises: associating the additional compressed boot data with the boot data list”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, and wherein the updating comprises: associating the additional compressed boot data with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Lillich discloses this limitation: *See* Claims 1.4.1, 2, 4, 5.6, and 8.6 above.	

98. The method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list.	Lillich, as evidenced by the example citations below, discloses "the method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 1.4.1, 5.6, and 8.6 above.	

107. The method of claim 2, further comprising: storing the updated boot list in a non-volatile memory.	Lillich, as evidenced by the example citations below, discloses "the method of claim 2, further comprising: storing the updated boot list in a non-volatile memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 2, further comprising: storing the updated boot list in a non-volatile memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 1.4.1, 5.6, and 8.6 above.	

108. The method of claim 2, further comprising: compressing at least a portion of the additional boot data.	Lillich, as evidenced by the example citations below, discloses "the method of claim 2, further comprising: compressing at least a portion of the additional boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 2, further comprising: compressing at least a portion of the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 1.1, 5, 9.1 and 8 above.	

109. The method of claim 108, further comprising: storing the compressed additional boot data.	Lillich, as evidenced by the example citations below, discloses "the method of claim 108, further comprising: storing the compressed additional boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 108, further comprising: storing the compressed additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lillich discloses this limitation: *See* Claims 5.1, 8.1 and 9.1 above.	

Appendix C3
Invalidity of U.S. Patent 8,880,862 based on Ballard

U.S. Patent No. 5,933,630 to Ballard ("Ballard") invalidates claims 1-6, 8-11, 13-17, 19, 23-24, 27-29, 31-33, 35-36, 39-40, 43-45, 47-53, 59-60, 63-65, 67, 71-72, 75-77, 79-81, 83-84, 87-89, 91-93, 97-98, and 107-109 of United States Patent No. 8,880,862 ("the '862 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '862 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix C3
Invalidity of U.S. Patent 8,880,862 based on Ballard

1 (Preamble) A method for providing accelerated loading of an operating system in a computer system, the method comprising:	Ballard, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, the method comprising:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system in a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

> Launch time for a computer program is reduced by logging hard disk accesses during an initial launch, then processing the log file to accelerate subsequent launches. The log file is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. The disk access log entries are sorted by physical address or are grouped by file, then organized by logical address within each group. The processed log file is stored with the application program. When the application program is launched thereafter, the processed log file is accessed first. All the disk accesses in the log file are performed moving all the data into RAM cache. When the program launch resumes, the launch occurs faster because all the data is already in cache.

Ballard, Abstract

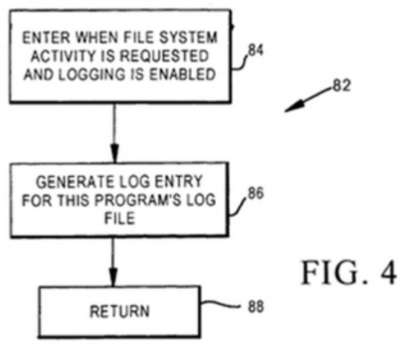
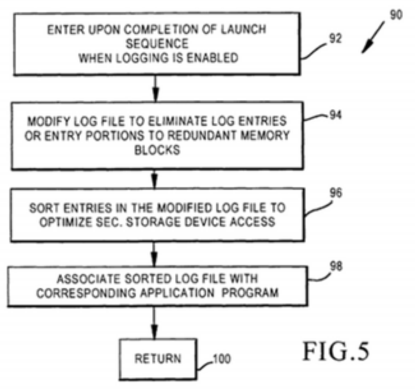

ENTER WHEN FILE SYSTEM ACTIVITY IS REQUESTED AND LOGGING IS ENABLED
84
82
GENERATE LOG ENTRY FOR THIS PROGRAM'S LOG FILE
86
RETURN
88

FIG. 4

Ballard, Fig. 4

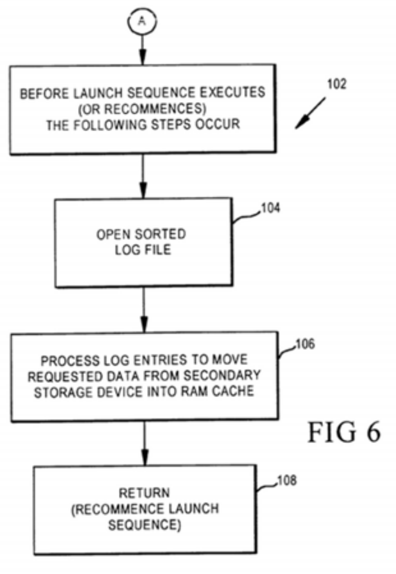
Appendix C3
Invalidity of U.S. Patent 8,880,862 based on Ballard



90
ENTER UPON COMPLETION OF LAUNCH SEQUENCE WHEN LOGGING IS ENABLED
92
MODIFY LOG FILE TO ELIMINATE LOG ENTRIES OR ENTRY PORTIONS TO REDUNDANT MEMORY BLOCKS
94
SORT ENTRIES IN THE MODIFIED LOG FILE TO OPTIMIZE SEC. STORAGE DEVICE ACCESS
96
ASSOCIATE SORTED LOG FILE WITH CORRESPONDING APPLICATION PROGRAM
98
RETURN
100

FIG.5

Ballard, Fig. 5



A
BEFORE LAUNCH SEQUENCE EXECUTES (OR RECOMMENCES) THE FOLLOWING STEPS OCCUR
102
OPEN SORTED LOG FILE
104
PROCESS LOG ENTRIES TO MOVE REQUESTED DATA FROM SECONDARY STORAGE DEVICE INTO RAM CACHE
106
RETURN (RECOMMENCE LAUNCH SEQUENCE)
108

FIG 6

Ballard, Fig. 6

1.1 loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory;	Ballard, as evidenced by the example citations below, discloses "loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

> Launch time for a computer program is reduced by logging hard disk accesses during an initial launch, then processing the log file to accelerate subsequent launches. The log file is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. The disk access log entries are sorted by physical address or are grouped by file, then organized by logical address within each group. The processed log file is stored with the application program. When the application program is launched thereafter, the processed log file is accessed first. All the disk accesses in the log file are performed moving all the data into RAM cache. When the program launch resumes, the launch occurs faster because all the data is already in cache.

Ballard, Abstract

> It is common for an application program for a personal computer to be stored in multiple files on the secondary storage device. There often is an executable file, a preferences file and many other files. Some programs include a data base file or a default data file. During a launch of the program multiple files are opened and select portions are moved from the secondary storage device into the primary storage device. When purchasing a computer program the packaging often specifies the amount of RAM address space (i.e., primary storage device address space) required to be allocated to the program while active. By active it is meant that the program has been launched and is currently processing or is

currently able to accept input commands.

Ballard, 1:51-63

> According to another aspect of the invention, the launch access log is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. According to one approach the disk access log entries are sorted by physical address. According to another approach the disk access log entries are grouped by file, then organized by logical address within each group. The processed log file then is stored with the application program.

Ballard, 2:22-30

> The primary storage device 14 typically is a storage device having a faster access time than that of the secondary storage device. An exemplary primary storage device 14 is random access memory (RAM). All or a portion of the RAM serves as a RAM cache 15. Portions of a computer program and/or data files are loaded into the RAM cache 15 to speed up execution of the program and processing of data. Mass produced computer software typically include specifications requiring a minimum amount of RAM required to run the program on a given computer system. During a launch sequence for starting such a computer program, portions of the program are copied from the secondary storage device into RAM.
>
> A launch sequence as used herein means the sequence of steps executed by the computer system during the launch time which pertain to starting up a given computer program and getting the computer ready to accept input commands for the computer program. A launch sequence is executed for a computer program. Different computer programs have different launch sequences. Steps included in a launch sequence include copying portions of the computer program being launched from the secondary storage device to the primary storage device. Other steps may include allocating a port or device to serve as an input source and/or output receptor. The method of this invention for accelerating a program launch is directed to improving the speed for accessing the secondary storage device during a launch sequence.

Ballard, 3:40-67

> The log file corresponds to a specific computer program--the one being launched that triggered such log file to be created. When multiple programs are being launched at the same time, a log file is created for each such program. The interrupt service routine then sets a flag at step 78 to indicate that logging is enabled for such program. At step 80 the

program returns. If the trigger for calling the routine 70 was for a secondary storage device access, then the steps at FIG. 4 also are performed before returning.

Log File System Activity

Once the launch of a computer program is detected and a log file is opened, all file system activity is monitored. Specifically, for each operating system call to the file system the call is analyzed to determine to which application being launched, if any, does the call pertain. Exemplary operating system calls to the file system are OPEN, READ, WRITE, and CLOSE. Referring to FIG. 4 routine 82 is entered at step 84 when both file system activity is detected and logging is enabled. If the call pertains to a computer program being launched, then an entry is appended to the appropriate log file (step 86). The routine 82 then returns at step 88.

The log entry includes a file identifier, the logical address(es) specified in the call and the time of access (e.g., system time; index value). Alternatively, the physical memory address(es) corresponding to the logical address(es) are stored in the log entry. In some embodiments, the operating system has already caused the physical addresses to be generated. If not, then the physical addresses are derived from the logical addresses using the operating system's file allocation table to translate the logical address into the physical address.

A logical address (also referred to as a virtual address) is the address which the computer program uses to access memory. A memory management unit translates this address into a physical address before the actual memory is read or written. A physical address is a memory location on the secondary storage device 16.

Ballard, 5:1-37

Detect Launch Completion

When completion of a launch sequence occurs and logging is enabled, then routine 90 is executed. Referring to FIG. 5, the routine enters at step 92. Conventional operating systems have a specific function that is called when a program is ready for normal execution. Under the Macintosh operating system, the function "Get Next Event" is called. For a computer program running under such operating system, such function is called by the computer program when the launch sequence is complete. According to one embodiment of this invention, step 92 is implemented by an interrupt service routine which is called whenever a computer program calls such function. The interrupt service routine clears the logging enabled flag for the corresponding application program.

In an alternative embodiment, access activity to the secondary storage device 16 is monitored to determine when activity has ceased for a threshold length of time (e.g., 3 seconds). Alternatively or in addition activity is monitored to determine when activity has gone below a threshold data throughput rate (e.g., 50 kilobytes per second) for a threshold period of time (e.g., 5 seconds). When there is insufficient activity for such threshold time, then the program launch is considered to be complete. The routine 90 then is executed.

Modify and Sort Log File

Once the launch sequence is determined to have been completed, then the log entry order is re-organized. The purpose is to eliminate redundant accesses to the same memory block and to optimize access time for the secondary storage device. If accesses occur faster, then the launch time (i.e., time elapsed from start to finish of launch sequence is less) is reduced. Thus, the launch of the computer program is accelerated.

Referring to FIG. 5, at step 94 the log file is processed to eliminate log entries or log entry portions to redundant memory blocks.

Ballard, 5:55-6:23

Note in this example that the log includes redundant entries. Entry h2 is a subset of entry d. Entry i2 is the same as entry f. Entry l2 is the same as entry a. It is expected that a redundant access request results from a computer program launch sequence access specifying a different address in the same block as previously accessed.

Frequently the log file will include an entry specifying a single address. Even though only one address is specified, an entire memory block will be accessed (read or written). This is because the minimum file allocation unit is a memory block cluster. Thus, the smallest portion of the computer program than can be accessed is the cluster size (i.e., cluster size). Entries in the log file are tested at step 94 to identify any redundant accesses to portions of the computer program. To determine whether a later entry is an access to a redundant portion of the computer program, the cluster address for the access is identified. The cluster address is either stored in the log file or is derived from the address stored in the log file. For a FAT drive the cluster size is stored in the drive's boot sector. The boot sector includes information about the layout of the file system used for the drive partition. Following the boot sector are several reserved sectors.

Following the reserved sectors is the file allocation table (FAT). Following the FAT are one or more backup copies of the FAT. Following the backup copies is the root directory. The size of the root directory is specified in the boot sector. After the root directory are the user's file and directory area. This is the area divided into clusters. Thus, from the information in the boot sector the starting address of cluster space is determined, along with the size of a cluster. Thus, the address boundaries for each cluster are known.

The address for any given cluster x is given by Ax+B, where A and B are constants determined from the boot partition. Thus, for any given file system call the cluster boundaries for such address are determinable. The log file stores either the accessed address, the cluster number (i.e., x) or the cluster address (e.g., start address, end address or some other identifying address) for each cluster accessed.

Ballard, 6:44-7:14

Following is a description of the redundancy testing. Consider the following access sequence: address 139, address 119, address 110, addresses 120-133, and addresses 138-140. Also consider that the memory block and cluster size is 10 addresses, and that blocks are located at 100-109, 110-119, 120-129, 130-139, 140-149. When address 139 is accessed the contents within the block of addresses 130-139 is accessed. For an embodiment which stores a starting address of the cluster as the cluster address, the log entry includes address 130. When address 119 is accessed the contents within the block of addresses 110-119 is accessed. The log entry for such access includes address 110. The next access in the launch sequence specifies address 110, causing the block 110-119 to be accessed. The cluster address is 110 which is already in the log. This access is a redundant access to a cluster already specified. The redundant access is eliminated by deleting the log entry for address 110. The next access specifies addresses 120-133. This access spans two clusters 120-129 and 130-139. The accesses are logged separately. The log entry for cluster address 120 is not redundant. The log entry for cluster address 130, however, is redundant because the first log entry for address 139 encompasses the memory block of addresses 130-139. To eliminate the redundancy, the log entry for address 139 is removed. The next access specifies addresses 139-140. This access also spans two clusters with two log entries. The first of the two is for cluster 130-139. This is a redundant entry and thus is removed from the log fie. The second of the two entries is for cluster 140-149. This is a nonredundant access. When the end of the log file is reached then redundancy testing (step 94) is complete.

In many instances the redundant accesses are eliminated from consideration by a cache operation performed by the computer instead of

> by step 94. Specifically, when caching is performed the second access to the same cluster will not result in a call to the hard drive because the data is in the cache. The cache will satisfy the request. Requests satisfied by the cache are ignored for purposes of creating a log of accesses. Thus, redundant entries do not get logged.

Ballard, 7:15-53.

See also Ballard, Tables B-C, Figs. 4-6

1.2 accessing the loaded portion of the boot data in the compressed form from memory;	Ballard, as evidenced by the example citations below, discloses “accessing the loaded portion of the boot data in the compressed form from memory;”

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded portion of the boot data in the compressed form from memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.

Ballard discloses this limitation:

> Launch time for a computer program is reduced by logging hard disk accesses during an initial launch, then processing the log file to accelerate subsequent launches. The log file is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. The disk access log entries are sorted by physical address or are grouped by file, then organized by logical address within each group. The processed log file is stored with the application program. When the application program is launched thereafter, the processed log file is accessed first. All the disk accesses in the log file are performed moving all the data into RAM cache. When the program launch resumes, the launch occurs faster because all the data is already in cache.

Ballard, Abstract

> A general purpose personal computer typically has many interactive application computer programs installed. A user is able to start-up multiple programs. With regard to an interactive computer program, the term "launch time", as used herein, means the time from when a processor receives a command to start the computer program until the time that the computer program is ready to accept input commands (e.g., user interface commands, batch-entry commands). The term "launch" as used herein means the process performed during the launch time to start up the computer program and get the computer ready to accept input commands for the computer progra[m].

Ballard, 1:38-50

> According to another aspect of the invention, the launch access log is

> processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. According to one approach the disk access log entries are sorted by physical address. According to another approach the disk access log entries are grouped by file, then organized by logical address within each group. The processed log file then is stored with the application program.

Ballard, 2:22-30

> The primary storage device 14 typically is a storage device having a faster access time than that of the secondary storage device. An exemplary primary storage device 14 is random access memory (RAM). All or a portion of the RAM serves as a RAM cache 15. Portions of a computer program and/or data files are loaded into the RAM cache 15 to speed up execution of the program and processing of data. Mass produced computer software typically include specifications requiring a minimum amount of RAM required to run the program on a given computer system. During a launch sequence for starting such a computer program, portions of the program are copied from the secondary storage device into RAM.
>
> A launch sequence as used herein means the sequence of steps executed by the computer system during the launch time which pertain to starting up a given computer program and getting the computer ready to accept input commands for the computer program. A launch sequence is executed for a computer program. Different computer programs have different launch sequences. Steps included in a launch sequence include copying portions of the computer program being launched from the secondary storage device to the primary storage device. Other steps may include allocating a port or device to serve as an input source and/or output receptor. The method of this invention for accelerating a program launch is directed to improving the speed for accessing the secondary storage device during a launch sequence.

Ballard, 3:40-67

See also Ballard, 5:1-7:53, Figs. 4-6

1.3 decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form; and	Ballard, as evidenced by the example citations below, discloses "decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

> Launch time for a computer program is reduced by logging hard disk accesses during an initial launch, then processing the log file to accelerate subsequent launches. The log file is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. The disk access log entries are sorted by physical address or are grouped by file, then organized by logical address within each group. The processed log file is stored with the application program. When the application program is launched thereafter, the processed log file is accessed first. All the disk accesses in the log file are performed moving all the data into RAM cache. When the program launch resumes, the launch occurs faster because all the data is already in cache.

Ballard, Abstract

> A general purpose personal computer typically has many interactive application computer programs installed. A user is able to start-up multiple programs. With regard to an interactive computer program, the term "launch time", as used herein, means the time from when a processor receives a command to start the computer program until the time that the computer program is ready to accept input commands (e.g., user interface commands, batch-entry commands). The term "launch" as used herein means the process performed during the launch time to start up the computer program and get the computer ready to accept input commands for the computer progra[m].

Ballard, 1:38-50

> According to another aspect of the invention, the launch access log is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. According to one approach the disk access log entries are sorted by physical address. According to another approach the disk access log entries are grouped by file, then organized by logical address within each group. The processed log file then is stored with the application program.

Ballard, 2:22-30

> The primary storage device 14 typically is a storage device having a faster access time than that of the secondary storage device. An exemplary primary storage device 14 is random access memory (RAM). All or a portion of the RAM serves as a RAM cache 15. Portions of a computer program and/or data files are loaded into the RAM cache 15 to speed up execution of the program and processing of data. Mass produced computer software typically include specifications requiring a minimum amount of RAM required to run the program on a given computer system. During a launch sequence for starting such a computer program, portions of the program are copied from the secondary storage device into RAM.
>
> A launch sequence as used herein means the sequence of steps executed by the computer system during the launch time which pertain to starting up a given computer program and getting the computer ready to accept input commands for the computer program. A launch sequence is executed for a computer program. Different computer programs have different launch sequences. Steps included in a launch sequence include copying portions of the computer program being launched from the secondary storage device to the primary storage device. Other steps may include allocating a port or device to serve as an input source and/or output receptor. The method of this invention for accelerating a program launch is directed to improving the speed for accessing the secondary storage device during a launch sequence.

Ballard, 3:40-67

See also Ballard, 5:1-7:53, Figs. 4-6

1.4.1 updating the boot data list,	Ballard, as evidenced by the example citations below, discloses "updating the boot data list,"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

> Launch time for a computer program is reduced by logging hard disk accesses during an initial launch, then processing the log file to accelerate subsequent launches. The log file is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. The disk access log entries are sorted by physical address or are grouped by file, then organized by logical address within each group. The processed log file is stored with the application program. When the application program is launched thereafter, the processed log file is accessed first. All the disk accesses in the log file are performed moving all the data into RAM cache. When the program launch resumes, the launch occurs faster because all the data is already in cache.

Ballard, Abstract

> According to another aspect of the invention, the launch access log is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. According to one approach the disk access log entries are sorted by physical address. According to another approach the disk access log entries are grouped by file, then organized by logical address within each group. The processed log file then is stored with the application program.

Ballard, 2:22-30

> Following is a description of the redundancy testing. Consider the following access sequence: address 139, address 119, address 110, addresses 120-133, and addresses 138-140. Also consider that the memory block and cluster size is 10 addresses, and that blocks are located at 100-109, 110-119, 120-129, 130-139, 140-149. When address 139 is accessed the contents within the block of addresses 130-139 is accessed. For an embodiment which stores a starting address of the cluster as the cluster address, the log entry includes address 130. When

address 119 is accessed the contents within the block of addresses 110-119 is accessed. The log entry for such access includes address 110. The next access in the launch sequence specifies address 110, causing the block 110-119 to be accessed. The cluster address is 110 which is already in the log. This access is a redundant access to a cluster already specified. The redundant access is eliminated by deleting the log entry for address 110. The next access specifies addresses 120-133. This access spans two clusters 120-129 and 130-139. The accesses are logged separately. The log entry for cluster address 120 is not redundant. The log entry for cluster address 130, however, is redundant because the first log entry for address 139 encompasses the memory block of addresses 130-139. To eliminate the redundancy, the log entry for address 139 is removed. The next access specifies addresses 139-140. This access also spans two clusters with two log entries. The first of the two is for cluster 130-139. This is a redundant entry and thus is removed from the log fie. The second of the two entries is for cluster 140-149. This is a nonredundant access. When the end of the log file is reached then redundancy testing (step 94) is complete.

In many instances the redundant accesses are eliminated from consideration by a cache operation performed by the computer instead of by step 94. Specifically, when caching is performed the second access to the same cluster will not result in a call to the hard drive because the data is in the cache. The cache will satisfy the request. Requests satisfied by the cache are ignored for purposes of creating a log of accesses. Thus, redundant entries do not get logged.

The modified log file next is sorted at step 96. In one embodiment the log entries are grouped according to the file. Each access resulting in a log entry specifies a file and an address. All entries for a given file are grouped together, then arranged within the group by logical address. The groups are arranged chronologically according to which file is specified first in the log file. Alternatively other criteria for ordering the groups is used. In an alternative embodiment instead of grouping the entries by file, all the log entries are sorted according to physical address to optimize access time to the secondary storage device 16. In one embodiment the log entries define a queue processed according to the methods disclosed in U.S. patent application Ser. No. 08/656,372 filed May 31, 1996 for "Estimating Access Time for Hard Drive I/O Requests." The contents of such application are incorporated herein by reference and made a part hereof. The log entries are rearranged in an order to optimize access time as described therein.

Ballard, 7:15-8:4.

See also Ballard, 5:1-7:14, Figs. 4-6

1.4.2 wherein the decompressed portion of boot data comprises a portion of the operating system.	Ballard, as evidenced by the example citations below, discloses "wherein the decompressed portion of boot data comprises a portion of the operating system."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the decompressed portion of boot data comprises a portion of the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

> The processor 12 serves to execute an operating system and one or more application computer programs. In some embodiments there are multiple processors for executing the operating system and application programs. System utilities and/or operating system extension programs also are executed according to some computer system 10 embodiments. Conventional operating systems include DOS, Windows, Windows NT, MacOS, OS/2 and various UNIX-based operating systems. The display device 18 and input devices 20 enable interaction between a user and the computer system 10. The computer system 10 in the process of executing the operating system and zero or more computer programs defines an operating environment for a user to interact with the computer, operating system and executing computer program. The display device 18 serves as an output device. Exemplary display devices include a CRT monitor or flat panel display. The user inputs commands and data to the computer system 10 via the input devices. Exemplary input devices include a keyboard, a pointing device and a clicking device. Data also is input to the computer via transportable disks or through I/O ports (not shown).

Ballard, 3:10-30.

> Log File System Activity
>
> Once the launch of a computer program is detected and a log file is opened, all file system activity is monitored. Specifically, for each operating system call to the file system the call is analyzed to determine to which application being launched, if any, does the call pertain. Exemplary operating system calls to the file system are OPEN, READ, WRITE, and CLOSE. Referring to FIG. 4 routine 82 is entered at step 84 when both file system activity is detected and logging is enabled. If the call pertains to a computer program being launched, then an entry is

1.4.2 wherein the decompressed portion of boot data comprises a portion of the operating system.	Ballard, as evidenced by the example citations below, discloses "wherein the decompressed portion of boot data comprises a portion of the operating system."
appended to the appropriate log file (step 86). The routine 82 then returns at step 88. The log entry includes a file identifier, the logical address(es) specified in the call and the time of access (e.g., system time; index value). Alternatively, the physical memory address(es) corresponding to the logical address(es) are stored in the log entry. In some embodiments, the operating system has already caused the physical addresses to be generated. If not, then the physical addresses are derived from the logical addresses using the operating system's file allocation table to translate the logical address into the physical address. A logical address (also referred to as a virtual address) is the address which the computer program uses to access memory. A memory management unit translates this address into a physical address before the actual memory is read or written. A physical address is a memory location on the secondary storage device 16. Ballard, 5:9-37 *See also* Ballard, 5:38-7:14, Figs. 4-6	

2. The method of claim 1, wherein the updating comprises: associating additional boot data with the boot data list.	Ballard, as evidenced by the example citations below, discloses "wherein the updating comprises: associating additional boot data with the boot data list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein associating additional boot data with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1.4.1 above. *See also* ***Look for additional references associating additional boot data with the existing boot data list.***	

3. The method of claim 1, wherein the updating comprises: removing an association of additional boot data that is associated with the boot data list from the boot data list.	Ballard, as evidenced by the example citations below, discloses "wherein the updating comprises: removing an association of additional boot data that is associated with the boot data list from the boot data list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein removing an association of additional boot data that is associated with the boot data list from the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1.4.1 above.	

4. The method of claim 1, wherein the updating comprises: associating additional boot data with the boot data list; and compressing a portion of the additional boot data.	Ballard, as evidenced by the example citations below, discloses "wherein the updating comprises: associating additional boot data with the boot data list; and compressing a portion of the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein associating additional boot data with the boot data list; and compressing a portion of the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 1.4.1, and 2 above.	

5 (Preamble) A method for booting a computer system, the method comprising:	Ballard, as evidenced by the example citations below, discloses "a method for booting a computer system, the method comprising:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

See Claims 1-1.4.2 above.

See also

> The log file corresponds to a specific computer program--the one being launched that triggered such log file to be created. When multiple programs are being launched at the same time, a log file is created for each such program. The interrupt service routine then sets a flag at step 78 to indicate that logging is enabled for such program. At step 80 the program returns. If the trigger for calling the routine 70 was for a secondary storage device access, then the steps at FIG. 4 also are performed before returning.
>
> Log File System Activity
>
> Once the launch of a computer program is detected and a log file is opened, all file system activity is monitored. Specifically, for each operating system call to the file system the call is analyzed to determine to which application being launched, if any, does the call pertain. Exemplary operating system calls to the file system are OPEN, READ, WRITE, and CLOSE. Referring to FIG. 4 routine 82 is entered at step 84 when both file system activity is detected and logging is enabled. If the call pertains to a computer program being launched, then an entry is appended to the appropriate log file (step 86). The routine 82 then returns at step 88.
>
> The log entry includes a file identifier, the logical address(es) specified in the call and the time of access (e.g., system time; index value). Alternatively, the physical memory address(es) corresponding to the logical address(es) are stored in the log entry. In some embodiments, the operating system has already caused the physical addresses to be

generated. If not, then the physical addresses are derived from the logical addresses using the operating system's file allocation table to translate the logical address into the physical address.

A logical address (also referred to as a virtual address) is the address which the computer program uses to access memory. A memory management unit translates this address into a physical address before the actual memory is read or written. A physical address is a memory location on the secondary storage device 16.

Ballard, 5:1-37

Detect Launch Completion

When completion of a launch sequence occurs and logging is enabled, then routine 90 is executed. Referring to FIG. 5, the routine enters at step 92. Conventional operating systems have a specific function that is called when a program is ready for normal execution. Under the Macintosh operating system, the function "Get Next Event" is called. For a computer program running under such operating system, such function is called by the computer program when the launch sequence is complete. According to one embodiment of this invention, step 92 is implemented by an interrupt service routine which is called whenever a computer program calls such function. The interrupt service routine clears the logging enabled flag for the corresponding application program.

In an alternative embodiment, access activity to the secondary storage device 16 is monitored to determine when activity has ceased for a threshold length of time (e.g., 3 seconds). Alternatively or in addition activity is monitored to determine when activity has gone below a threshold data throughput rate (e.g., 50 kilobytes per second) for a threshold period of time (e.g., 5 seconds). When there is insufficient activity for such threshold time, then the program launch is considered to be complete. The routine 90 then is executed.

Modify and Sort Log File

Once the launch sequence is determined to have been completed, then the log entry order is re-organized. The purpose is to eliminate redundant accesses to the same memory block and to optimize access time for the secondary storage device. If accesses occur faster, then the launch time (i.e., time elapsed from start to finish of launch sequence is less) is reduced. Thus, the launch of the computer program is accelerated.

Referring to FIG. 5, at step 94 the log file is processed to eliminate log

> entries or log entry portions to redundant memory blocks.

Ballard, 5:55-6:23

> Note in this example that the log includes redundant entries. Entry h2 is a subset of entry d. Entry i2 is the same as entry f. Entry l2 is the same as entry a. It is expected that a redundant access request results from a computer program launch sequence access specifying a different address in the same block as previously accessed.
>
> Frequently the log file will include an entry specifying a single address. Even though only one address is specified, an entire memory block will be accessed (read or written). This is because the minimum file allocation unit is a memory block cluster. Thus, the smallest portion of the computer program than can be accessed is the cluster size (i.e., cluster size). Entries in the log file are tested at step 94 to identify any redundant accesses to portions of the computer program. To determine whether a later entry is an access to a redundant portion of the computer program, the cluster address for the access is identified. The cluster address is either stored in the log file or is derived from the address stored in the log file. For a FAT drive the cluster size is stored in the drive's boot sector. The boot sector includes information about the layout of the file system used for the drive partition. Following the boot sector are several reserved sectors. Following the reserved sectors is the file allocation table (FAT). Following the FAT are one or more backup copies of the FAT. Following the backup copies is the root directory. The size of the root directory is specified in the boot sector. After the root directory are the user's file and directory area. This is the area divided into clusters. Thus, from the information in the boot sector the starting address of cluster space is determined, along with the size of a cluster. Thus, the address boundaries for each cluster are known.
>
> The address for any given cluster x is given by Ax+B, where A and B are constants determined from the boot partition. Thus, for any given file system call the cluster boundaries for such address are determinable. The log file stores either the accessed address, the cluster number (i.e., x) or the cluster address (e.g., start address, end address or some other identifying address) for each cluster accessed.

Ballard, 6:44-7:14

> Following is a description of the redundancy testing. Consider the following access sequence: address 139, address 119, address 110, addresses 120-133, and addresses 138-140. Also consider that the memory block and cluster size is 10 addresses, and that blocks are located

at 100-109, 110-119, 120-129, 130-139, 140-149. When address 139 is accessed the contents within the block of addresses 130-139 is accessed. For an embodiment which stores a starting address of the cluster as the cluster address, the log entry includes address 130. When address 119 is accessed the contents within the block of addresses 110-119 is accessed. The log entry for such access includes address 110. The next access in the launch sequence specifies address 110, causing the block 110-119 to be accessed. The cluster address is 110 which is already in the log. This access is a redundant access to a cluster already specified. The redundant access is eliminated by deleting the log entry for address 110. The next access specifies addresses 120-133. This access spans two clusters 120-129 and 130-139. The accesses are logged separately. The log entry for cluster address 120 is not redundant. The log entry for cluster address 130, however, is redundant because the first log entry for address 139 encompasses the memory block of addresses 130-139. To eliminate the redundancy, the log entry for address 139 is removed. The next access specifies addresses 139-140. This access also spans two clusters with two log entries. The first of the two is for cluster 130-139. This is a redundant entry and thus is removed from the log fie. The second of the two entries is for cluster 140-149. This is a nonredundant access. When the end of the log file is reached then redundancy testing (step 94) is complete.

In many instances the redundant accesses are eliminated from consideration by a cache operation performed by the computer instead of by step 94. Specifically, when caching is performed the second access to the same cluster will not result in a call to the hard drive because the data is in the cache. The cache will satisfy the request. Requests satisfied by the cache are ignored for purposes of creating a log of accesses. Thus, redundant entries do not get logged.

Ballard, 7:15-53.

See also Ballard, Abstract, Tables B-C, Figs. 4-6.

5.1 storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory;	Ballard, as evidenced by the example citations below, discloses "storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

See Claim 1.1 above.

See also

> The secondary storage device 16 serves as a permanent storage memory for one or more computer programs 24 to be executed by the processor 12. The secondary storage device 16 also stores data files for use with the application computer programs. Exemplary secondary storage devices include a hard disk drive, floppy disk drive, CD-ROM drive, bernoulli disk drive or other drive system for accessing permanent or replaceable disks, such as floppy disks, magnetic disks, magneto-optical disks, or optical disks.
>
> The primary storage device 14 typically is a storage device having a faster access time than that of the secondary storage device. An exemplary primary storage device 14 is random access memory (RAM). All or a portion of the RAM serves as a RAM cache 15. Portions of a computer program and/or data files are loaded into the RAM cache 15 to speed up execution of the program and processing of data. Mass produced computer software typically include specifications requiring a minimum amount of RAM required to run the program on a given computer system. During a launch sequence for starting such a computer program, portions of the program are copied from the secondary storage device into RAM.

Ballard, 3:31-52

5.2 loading the stored compressed boot data from the first memory;	Ballard, as evidenced by the example citations below, discloses “loading the stored compressed boot data from the first memory”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the stored compressed boot data from the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1.1 above.	

5.3 accessing the loaded compressed boot data;	Ballard, as evidenced by the example citations below, discloses "accessing the loaded compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1.2 above.	

5.4 decompressing the accessed compressed boot data;	Ballard, as evidenced by the example citations below, discloses "decompressing the accessed compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1.3 above.	

5.5 utilizing the decompressed boot data to at least partially boot the computer system;	Ballard, as evidenced by the example citations below, discloses "utilizing the decompressed boot data to at least partially boot the computer system"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed boot data to at least partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

See Claims 1.3 and 1.4.2 above.

See also

> Launch time for a computer program is reduced by logging hard disk accesses during an initial launch, then processing the log file to accelerate subsequent launches. The log file is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. The disk access log entries are sorted by physical address or are grouped by file, then organized by logical address within each group. The processed log file is stored with the application program. When the application program is launched thereafter, the processed log file is accessed first. All the disk accesses in the log file are performed moving all the data into RAM cache. When the program launch resumes, the launch occurs faster because all the data is already in cache.

Ballard, Abstract

> It is common for an application program for a personal computer to be stored in multiple files on the secondary storage device. There often is an executable file, a preferences file and many other files. Some programs include a data base file or a default data file. During a launch of the program multiple files are opened and select portions are moved from the secondary storage device into the primary storage device. When purchasing a computer program the packaging often specifies the amount of RAM address space (i.e., primary storage device address space) required to be allocated to the program while active. By active it is meant that the program has been launched and is currently processing or is

> currently able to accept input commands.

Ballard, 1:51-63

> According to another aspect of the invention, the launch access log is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. According to one approach the disk access log entries are sorted by physical address. According to another approach the disk access log entries are grouped by file, then organized by logical address within each group. The processed log file then is stored with the application program.

Ballard, 2:22-30

> The primary storage device 14 typically is a storage device having a faster access time than that of the secondary storage device. An exemplary primary storage device 14 is random access memory (RAM). All or a portion of the RAM serves as a RAM cache 15. Portions of a computer program and/or data files are loaded into the RAM cache 15 to speed up execution of the program and processing of data. Mass produced computer software typically include specifications requiring a minimum amount of RAM required to run the program on a given computer system. During a launch sequence for starting such a computer program, portions of the program are copied from the secondary storage device into RAM.
>
> A launch sequence as used herein means the sequence of steps executed by the computer system during the launch time which pertain to starting up a given computer program and getting the computer ready to accept input commands for the computer program. A launch sequence is executed for a computer program. Different computer programs have different launch sequences. Steps included in a launch sequence include copying portions of the computer program being launched from the secondary storage device to the primary storage device. Other steps may include allocating a port or device to serve as an input source and/or output receptor. The method of this invention for accelerating a program launch is directed to improving the speed for accessing the secondary storage device during a launch sequence.

Ballard, 3:40-67

> Following is a description of the redundancy testing. Consider the following access sequence: address 139, address 119, address 110, addresses 120-133, and addresses 138-140. Also consider that the memory block and cluster size is 10 addresses, and that blocks are located at 100-109, 110-119, 120-129, 130-139, 140-149. When address

139 is accessed the contents within the block of addresses 130-139 is accessed. For an embodiment which stores a starting address of the cluster as the cluster address, the log entry includes address 130. When address 119 is accessed the contents within the block of addresses 110-119 is accessed. The log entry for such access includes address 110. The next access in the launch sequence specifies address 110, causing the block 110-119 to be accessed. The cluster address is 110 which is already in the log. This access is a redundant access to a cluster already specified. The redundant access is eliminated by deleting the log entry for address 110. The next access specifies addresses 120-133. This access spans two clusters 120-129 and 130-139. The accesses are logged separately. The log entry for cluster address 120 is not redundant. The log entry for cluster address 130, however, is redundant because the first log entry for address 139 encompasses the memory block of addresses 130-139. To eliminate the redundancy, the log entry for address 139 is removed. The next access specifies addresses 139-140. This access also spans two clusters with two log entries. The first of the two is for cluster 130-139. This is a redundant entry and thus is removed from the log fie. The second of the two entries is for cluster 140-149. This is a nonredundant access. When the end of the log file is reached then redundancy testing (step 94) is complete.

In many instances the redundant accesses are eliminated from consideration by a cache operation performed by the computer instead of by step 94. Specifically, when caching is performed the second access to the same cluster will not result in a call to the hard drive because the data is in the cache. The cache will satisfy the request. Requests satisfied by the cache are ignored for purposes of creating a log of accesses. Thus, redundant entries do not get logged.

Ballard, 7:15-53.

5.6 updating the boot data list;	Ballard, as evidenced by the example citations below, discloses “updating the boot data list”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1.4.1 above.	

5.7 wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form.	Ballard, as evidenced by the example citations below, discloses "wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

See Claims 1-1.3 above.

See also

> Launch time for a computer program is reduced by logging hard disk accesses during an initial launch, then processing the log file to accelerate subsequent launches. The log file is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. The disk access log entries are sorted by physical address or are grouped by file, then organized by logical address within each group. The processed log file is stored with the application program. When the application program is launched thereafter, the processed log file is accessed first. All the disk accesses in the log file are performed moving all the data into RAM cache. When the program launch resumes, the launch occurs faster because all the data is already in cache.

Ballard, Abstract

> A general purpose personal computer typically has many interactive application computer programs installed. A user is able to start-up multiple programs. With regard to an interactive computer program, the term "launch time", as used herein, means the time from when a processor receives a command to start the computer program until the time that the computer program is ready to accept input commands (e.g., user interface

> commands, batch-entry commands). The term "launch" as used herein means the process performed during the launch time to start up the computer program and get the computer ready to accept input commands for the computer progra[m].

Ballard, 1:38-50

> According to another aspect of the invention, the launch access log is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. According to one approach the disk access log entries are sorted by physical address. According to another approach the disk access log entries are grouped by file, then organized by logical address within each group. The processed log file then is stored with the application program.

Ballard, 2:22-30

> The primary storage device 14 typically is a storage device having a faster access time than that of the secondary storage device. An exemplary primary storage device 14 is random access memory (RAM). All or a portion of the RAM serves as a RAM cache 15. Portions of a computer program and/or data files are loaded into the RAM cache 15 to speed up execution of the program and processing of data. Mass produced computer software typically include specifications requiring a minimum amount of RAM required to run the program on a given computer system. During a launch sequence for starting such a computer program, portions of the program are copied from the secondary storage device into RAM.
>
> A launch sequence as used herein means the sequence of steps executed by the computer system during the launch time which pertain to starting up a given computer program and getting the computer ready to accept input commands for the computer program. A launch sequence is executed for a computer program. Different computer programs have different launch sequences. Steps included in a launch sequence include copying portions of the computer program being launched from the secondary storage device to the primary storage device. Other steps may include allocating a port or device to serve as an input source and/or output receptor. The method of this invention for accelerating a program launch is directed to improving the speed for accessing the secondary storage device during a launch sequence.

Ballard, 3:40-67

> Following is a description of the redundancy testing. Consider the following access sequence: address 139, address 119, address 110,

addresses 120-133, and addresses 138-140. Also consider that the memory block and cluster size is 10 addresses, and that blocks are located at 100-109, 110-119, 120-129, 130-139, 140-149. When address 139 is accessed the contents within the block of addresses 130-139 is accessed. For an embodiment which stores a starting address of the cluster as the cluster address, the log entry includes address 130. When address 119 is accessed the contents within the block of addresses 110-119 is accessed. The log entry for such access includes address 110. The next access in the launch sequence specifies address 110, causing the block 110-119 to be accessed. The cluster address is 110 which is already in the log. This access is a redundant access to a cluster already specified. The redundant access is eliminated by deleting the log entry for address 110. The next access specifies addresses 120-133. This access spans two clusters 120-129 and 130-139. The accesses are logged separately. The log entry for cluster address 120 is not redundant. The log entry for cluster address 130, however, is redundant because the first log entry for address 139 encompasses the memory block of addresses 130-139. To eliminate the redundancy, the log entry for address 139 is removed. The next access specifies addresses 139-140. This access also spans two clusters with two log entries. The first of the two is for cluster 130-139. This is a redundant entry and thus is removed from the log fie. The second of the two entries is for cluster 140-149. This is a nonredundant access. When the end of the log file is reached then redundancy testing (step 94) is complete.

In many instances the redundant accesses are eliminated from consideration by a cache operation performed by the computer instead of by step 94. Specifically, when caching is performed the second access to the same cluster will not result in a call to the hard drive because the data is in the cache. The cache will satisfy the request. Requests satisfied by the cache are ignored for purposes of creating a log of accesses. Thus, redundant entries do not get logged.

Ballard, 7:15-53.

Appendix C3
Invalidity of U.S. Patent 8,880,862 based on Ballard

6 (Preamble) A system comprising: a processor;	Ballard, as evidenced by the example citations below, discloses "a system comprising: a processor:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system comprising: a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

> The log file corresponds to a specific computer program--the one being launched that triggered such log file to be created. When multiple programs are being launched at the same time, a log file is created for each such program. The interrupt service routine then sets a flag at step 78 to indicate that logging is enabled for such program. At step 80 the program returns. If the trigger for calling the routine 70 was for a secondary storage device access, then the steps at FIG. 4 also are performed before returning.
>
> Log File System Activity
>
> Once the launch of a computer program is detected and a log file is opened, all file system activity is monitored. Specifically, for each operating system call to the file system the call is analyzed to determine to which application being launched, if any, does the call pertain. Exemplary operating system calls to the file system are OPEN, READ, WRITE, and CLOSE. Referring to FIG. 4 routine 82 is entered at step 84 when both file system activity is detected and logging is enabled. If the call pertains to a computer program being launched, then an entry is appended to the appropriate log file (step 86). The routine 82 then returns at step 88.
>
> The log entry includes a file identifier, the logical address(es) specified in the call and the time of access (e.g., system time; index value). Alternatively, the physical memory address(es) corresponding to the logical address(es) are stored in the log entry. In some embodiments, the operating system has already caused the physical addresses to be generated. If not, then the physical addresses are derived from the logical addresses using the operating system's file allocation table to translate the logical address into the physical address.
>
> A logical address (also referred to as a virtual address) is the address which the computer program uses to access memory. A memory

management unit translates this address into a physical address before the actual memory is read or written. A physical address is a memory location on the secondary storage device 16.

Ballard, 5:1-37

Detect Launch Completion

When completion of a launch sequence occurs and logging is enabled, then routine 90 is executed. Referring to FIG. 5, the routine enters at step 92. Conventional operating systems have a specific function that is called when a program is ready for normal execution. Under the Macintosh operating system, the function "Get Next Event" is called. For a computer program running under such operating system, such function is called by the computer program when the launch sequence is complete. According to one embodiment of this invention, step 92 is implemented by an interrupt service routine which is called whenever a computer program calls such function. The interrupt service routine clears the logging enabled flag for the corresponding application program.

In an alternative embodiment, access activity to the secondary storage device 16 is monitored to determine when activity has ceased for a threshold length of time (e.g., 3 seconds). Alternatively or in addition activity is monitored to determine when activity has gone below a threshold data throughput rate (e.g., 50 kilobytes per second) for a threshold period of time (e.g., 5 seconds). When there is insufficient activity for such threshold time, then the program launch is considered to be complete. The routine 90 then is executed.

Modify and Sort Log File

Once the launch sequence is determined to have been completed, then the log entry order is re-organized. The purpose is to eliminate redundant accesses to the same memory block and to optimize access time for the secondary storage device. If accesses occur faster, then the launch time (i.e., time elapsed from start to finish of launch sequence is less) is reduced. Thus, the launch of the computer program is accelerated.

Referring to FIG. 5, at step 94 the log file is processed to eliminate log entries or log entry portions to redundant memory blocks.

Ballard, 5:55-6:23

Note in this example that the log includes redundant entries. Entry h2 is a subset of entry d. Entry i2 is the same as entry f. Entry l2 is the same as entry a. It is expected that a redundant access request results from a computer program launch sequence access specifying a different address in the same block as previously accessed.

Frequently the log file will include an entry specifying a single address. Even though only one address is specified, an entire memory block will be accessed (read or written). This is because the minimum file allocation unit is a memory block cluster. Thus, the smallest portion of the computer program than can be accessed is the cluster size (i.e., cluster size). Entries in the log file are tested at step 94 to identify any redundant accesses to portions of the computer program. To determine whether a later entry is an access to a redundant portion of the computer program, the cluster address for the access is identified. The cluster address is either stored in the log file or is derived from the address stored in the log file. For a FAT drive the cluster size is stored in the drive's boot sector. The boot sector includes information about the layout of the file system used for the drive partition. Following the boot sector are several reserved sectors. Following the reserved sectors is the file allocation table (FAT). Following the FAT are one or more backup copies of the FAT. Following the backup copies is the root directory. The size of the root directory is specified in the boot sector. After the root directory are the user's file and directory area. This is the area divided into clusters. Thus, from the information in the boot sector the starting address of cluster space is determined, along with the size of a cluster. Thus, the address boundaries for each cluster are known.

The address for any given cluster x is given by Ax+B, where A and B are constants determined from the boot partition. Thus, for any given file system call the cluster boundaries for such address are determinable. The log file stores either the accessed address, the cluster number (i.e., x) or the cluster address (e.g., start address, end address or some other identifying address) for each cluster accessed.

Ballard, 6:44-7:14

Following is a description of the redundancy testing. Consider the following access sequence: address 139, address 119, address 110, addresses 120-133, and addresses 138-140. Also consider that the memory block and cluster size is 10 addresses, and that blocks are located at 100-109, 110-119, 120-129, 130-139, 140-149. When address 139 is accessed the contents within the block of addresses 130-139 is accessed. For an embodiment which stores a starting address of the cluster as the cluster address, the log entry includes address 130. When address 119 is accessed the contents within the block of addresses 110-119 is accessed. The log entry for such access includes address 110. The next access in the launch sequence specifies address 110, causing the block 110-119 to be accessed. The cluster address is 110 which is already in the log. This access is a redundant access to a cluster already specified. The redundant access is eliminated by deleting the log entry for address 110. The next access specifies addresses 120-133. This access spans two clusters 120-129 and 130-139. The accesses are logged separately. The log entry for cluster address 120 is not redundant. The log entry for cluster address 130, however, is redundant because the first log entry for address 139 encompasses the memory block of addresses 130-139. To eliminate the redundancy, the log entry for address 139 is removed. The next access specifies addresses 139-140. This access also spans two clusters with two log entries. The first of the two is for cluster 130-139. This is a redundant entry and thus is removed from the log fie. The second of the two entries is for cluster 140-149. This is a nonredundant access. When the end of the log file is reached then redundancy testing (step 94) is complete.

In many instances the redundant accesses are eliminated from consideration by a cache operation performed by the computer instead of by step 94. Specifically, when caching is performed the second access to the same cluster will not result in a call to the hard drive because the data is in the cache. The cache will satisfy the request. Requests satisfied by the cache are ignored for purposes of creating a log of accesses. Thus, redundant entries do not get logged.

Ballard, 7:15-53.

See also Ballard, Abstract, Tables B-C, Figs. 4-6.

6.1 a memory; and	Ballard, as evidenced by the example citations below, discloses "a memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a memory.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 1.1 and 1.2 above.	

6.2 a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor	Ballard, as evidenced by the example citations below, discloses "a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

See Claims 1.1, 1.2, and 6(Preamble) above.

See also

> It is common for an application program for a personal computer to be stored in multiple files on the secondary storage device. There often is an executable file, a preferences file and many other files. Some programs include a data base file or a default data file. During a launch of the program multiple files are opened and select portions are moved from the secondary storage device into the primary storage device. When purchasing a computer program the packaging often specifies the amount of RAM address space (i.e., primary storage device address space) required to be allocated to the program while active. By active it is meant that the program has been launched and is currently processing or is currently able to accept input commands.

Ballard, 1:51-63

> According to another aspect of the invention, after program launch is complete the launch access log is processed. During a program's launch sequence multiple files are open at a given time. Contents from a first file are accessed, then contents from a second file. As the accesses continue the first file again is accessed. Thus, accesses to the multiple files are interspersed amongst each other. An exemplary sequence might be: File 1 (address 1), File 2 (address 20), File 3 (address 40), File 1 (address 6), File 3 (address 35), File 2 (address (25).

Ballard, 2:11-21

> The secondary storage device 16 serves as a permanent storage memory for one or more computer programs 24 to be executed by the processor 12. The secondary storage device 16 also stores data files for use with the application computer programs. Exemplary secondary storage devices include a hard disk drive, floppy disk drive, CD-ROM drive, bernoulli disk drive or other drive system for accessing permanent or replaceable disks, such as floppy disks, magnetic disks, magneto-optical disks, or optical disks.
>
> The primary storage device 14 typically is a storage device having a faster access time than that of the secondary storage device. An exemplary primary storage device 14 is random access memory (RAM). All or a portion of the RAM serves as a RAM cache 15. Portions of a computer program and/or data files are loaded into the RAM cache 15 to speed up execution of the program and processing of data. Mass produced computer software typically include specifications requiring a minimum amount of RAM required to run the program on a given computer system. During a launch sequence for starting such a computer program, portions of the program are copied from the secondary storage device into RAM.

Ballard, 3:31-52

6.3 wherein the processor is configured:	Ballard, as evidenced by the example citations below, discloses "wherein the processor is configured:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the processor is configured), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 6(Preamble) above	

6.4 to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory,	Ballard, as evidenced by the example citations below, discloses "to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1.1 above.	

6.5 to access the loaded portion of the boot data in the compressed form,	Ballard, as evidenced by the example citations below, discloses "to access the loaded portion of the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to access the loaded portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1.2 above.	

6.6 to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data, and	Ballard, as evidenced by the example citations below, discloses "to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1.3 above.	

6.7 to update the boot data list.	Ballard, as evidenced by the example citations below, discloses "to update the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to update the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1.4.1 above.	

Appendix C3
Invalidity of U.S. Patent 8,880,862 based on Ballard

8 (Preamble) A method of loading an operating system for booting a computer system, comprising:	Ballard, as evidenced by the example citations below, discloses "A method of loading an operating system for booting a computer system, comprising:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method of loading an operating system for booting a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1 (Preamble) above.	

8.1 storing a portion of the operating system in a compressed form in a first memory;	Ballard, as evidenced by the example citations below, discloses "storing a portion of the operating system in a compressed form in a first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing a portion of the operating system in a compressed form in a first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 6.2 above.	

8.2 loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list;	Ballard, as evidenced by the example citations below, discloses "loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1.1 above.	

8.3 accessing the loaded portion of the operating system from the second memory in the compressed form;	Ballard, as evidenced by the example citations below, discloses "accessing the loaded portion of the operating system from the second memory in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded portion of the operating system from the second memory in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1.2 above.	

8.4 decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system;	Ballard, as evidenced by the example citations below, discloses "decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 1.3 and 1.4.2 above.	

8.5 utilizing the decompressed portion of the operating system to at least partially boot the computer system; and	Ballard, as evidenced by the example citations below, discloses "utilizing the decompressed portion of the operating system to at least partially boot the computer system"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed portion of the operating system to at least partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

See Claims 1.3 and 1.4.2 above.

See also

> Launch time for a computer program is reduced by logging hard disk accesses during an initial launch, then processing the log file to accelerate subsequent launches. The log file is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. The disk access log entries are sorted by physical address or are grouped by file, then organized by logical address within each group. The processed log file is stored with the application program. When the application program is launched thereafter, the processed log file is accessed first. All the disk accesses in the log file are performed moving all the data into RAM cache. When the program launch resumes, the launch occurs faster because all the data is already in cache.

Ballard, Abstract

> It is common for an application program for a personal computer to be stored in multiple files on the secondary storage device. There often is an executable file, a preferences file and many other files. Some programs include a data base file or a default data file. During a launch of the program multiple files are opened and select portions are moved from the secondary storage device into the primary storage device. When purchasing a computer program the packaging often specifies the amount of RAM address space (i.e., primary storage device address space) required to be allocated to the program while active. By active it is meant

> that the program has been launched and is currently processing or is currently able to accept input commands.

Ballard, 1:51-63

> According to another aspect of the invention, the launch access log is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. According to one approach the disk access log entries are sorted by physical address. According to another approach the disk access log entries are grouped by file, then organized by logical address within each group. The processed log file then is stored with the application program.

Ballard, 2:22-30

> The primary storage device 14 typically is a storage device having a faster access time than that of the secondary storage device. An exemplary primary storage device 14 is random access memory (RAM). All or a portion of the RAM serves as a RAM cache 15. Portions of a computer program and/or data files are loaded into the RAM cache 15 to speed up execution of the program and processing of data. Mass produced computer software typically include specifications requiring a minimum amount of RAM required to run the program on a given computer system. During a launch sequence for starting such a computer program, portions of the program are copied from the secondary storage device into RAM.
>
> A launch sequence as used herein means the sequence of steps executed by the computer system during the launch time which pertain to starting up a given computer program and getting the computer ready to accept input commands for the computer program. A launch sequence is executed for a computer program. Different computer programs have different launch sequences. Steps included in a launch sequence include copying portions of the computer program being launched from the secondary storage device to the primary storage device. Other steps may include allocating a port or device to serve as an input source and/or output receptor. The method of this invention for accelerating a program launch is directed to improving the speed for accessing the secondary storage device during a launch sequence.

Ballard, 3:40-67

> Following is a description of the redundancy testing. Consider the following access sequence: address 139, address 119, address 110, addresses 120-133, and addresses 138-140. Also consider that the

memory block and cluster size is 10 addresses, and that blocks are located at 100-109, 110-119, 120-129, 130-139, 140-149. When address 139 is accessed the contents within the block of addresses 130-139 is accessed. For an embodiment which stores a starting address of the cluster as the cluster address, the log entry includes address 130. When address 119 is accessed the contents within the block of addresses 110-119 is accessed. The log entry for such access includes address 110. The next access in the launch sequence specifies address 110, causing the block 110-119 to be accessed. The cluster address is 110 which is already in the log. This access is a redundant access to a cluster already specified. The redundant access is eliminated by deleting the log entry for address 110. The next access specifies addresses 120-133. This access spans two clusters 120-129 and 130-139. The accesses are logged separately. The log entry for cluster address 120 is not redundant. The log entry for cluster address 130, however, is redundant because the first log entry for address 139 encompasses the memory block of addresses 130-139. To eliminate the redundancy, the log entry for address 139 is removed. The next access specifies addresses 139-140. This access also spans two clusters with two log entries. The first of the two is for cluster 130-139. This is a redundant entry and thus is removed from the log fie. The second of the two entries is for cluster 140-149. This is a nonredundant access. When the end of the log file is reached then redundancy testing (step 94) is complete.

In many instances the redundant accesses are eliminated from consideration by a cache operation performed by the computer instead of by step 94. Specifically, when caching is performed the second access to the same cluster will not result in a call to the hard drive because the data is in the cache. The cache will satisfy the request. Requests satisfied by the cache are ignored for purposes of creating a log of accesses. Thus, redundant entries do not get logged.

Ballard, 7:15-53.

8.6 updating the boot data list,	Ballard, as evidenced by the example citations below, discloses “updating the boot data list”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1.4.1 above.	

8.7 wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form.	Ballard, as evidenced by the example citations below, discloses "wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

See Claims 1-1.3 and 5.7 above.

See also

> Launch time for a computer program is reduced by logging hard disk accesses during an initial launch, then processing the log file to accelerate subsequent launches. The log file is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. The disk access log entries are sorted by physical address or are grouped by file, then organized by logical address within each group. The processed log file is stored with the application program. When the application program is launched thereafter, the processed log file is accessed first. All the disk accesses in the log file are performed moving all the data into RAM cache. When the program launch resumes, the launch occurs faster because all the data is already in cache.

Ballard, Abstract

> A general purpose personal computer typically has many interactive application computer programs installed. A user is able to start-up

> multiple programs. With regard to an interactive computer program, the term "launch time", as used herein, means the time from when a processor receives a command to start the computer program until the time that the computer program is ready to accept input commands (e.g., user interface commands, batch-entry commands). The term "launch" as used herein means the process performed during the launch time to start up the computer program and get the computer ready to accept input commands for the computer progra[m].

Ballard, 1:38-50

> According to another aspect of the invention, the launch access log is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. According to one approach the disk access log entries are sorted by physical address. According to another approach the disk access log entries are grouped by file, then organized by logical address within each group. The processed log file then is stored with the application program.

Ballard, 2:22-30

> The primary storage device 14 typically is a storage device having a faster access time than that of the secondary storage device. An exemplary primary storage device 14 is random access memory (RAM). All or a portion of the RAM serves as a RAM cache 15. Portions of a computer program and/or data files are loaded into the RAM cache 15 to speed up execution of the program and processing of data. Mass produced computer software typically include specifications requiring a minimum amount of RAM required to run the program on a given computer system. During a launch sequence for starting such a computer program, portions of the program are copied from the secondary storage device into RAM.
>
> A launch sequence as used herein means the sequence of steps executed by the computer system during the launch time which pertain to starting up a given computer program and getting the computer ready to accept input commands for the computer program. A launch sequence is executed for a computer program. Different computer programs have different launch sequences. Steps included in a launch sequence include copying portions of the computer program being launched from the secondary storage device to the primary storage device. Other steps may include allocating a port or device to serve as an input source and/or output receptor. The method of this invention for accelerating a program launch is directed to improving the speed for accessing the secondary

storage device during a launch sequence.

Ballard, 3:40-67

> Following is a description of the redundancy testing. Consider the following access sequence: address 139, address 119, address 110, addresses 120-133, and addresses 138-140. Also consider that the memory block and cluster size is 10 addresses, and that blocks are located at 100-109, 110-119, 120-129, 130-139, 140-149. When address 139 is accessed the contents within the block of addresses 130-139 is accessed. For an embodiment which stores a starting address of the cluster as the cluster address, the log entry includes address 130. When address 119 is accessed the contents within the block of addresses 110-119 is accessed. The log entry for such access includes address 110. The next access in the launch sequence specifies address 110, causing the block 110-119 to be accessed. The cluster address is 110 which is already in the log. This access is a redundant access to a cluster already specified. The redundant access is eliminated by deleting the log entry for address 110. The next access specifies addresses 120-133. This access spans two clusters 120-129 and 130-139. The accesses are logged separately. The log entry for cluster address 120 is not redundant. The log entry for cluster address 130, however, is redundant because the first log entry for address 139 encompasses the memory block of addresses 130-139. To eliminate the redundancy, the log entry for address 139 is removed. The next access specifies addresses 139-140. This access also spans two clusters with two log entries. The first of the two is for cluster 130-139. This is a redundant entry and thus is removed from the log fie. The second of the two entries is for cluster 140-149. This is a nonredundant access. When the end of the log file is reached then redundancy testing (step 94) is complete.
>
> In many instances the redundant accesses are eliminated from consideration by a cache operation performed by the computer instead of by step 94. Specifically, when caching is performed the second access to the same cluster will not result in a call to the hard drive because the data is in the cache. The cache will satisfy the request. Requests satisfied by the cache are ignored for purposes of creating a log of accesses. Thus, redundant entries do not get logged.

Ballard, 7:15-53.

9.1 The method of claim 8, further comprising: compressing an additional portion of the operating system that is not associated with the boot data list; and	Ballard, as evidenced by the example citations below, discloses "the method of claim 8, further comprising: compressing an additional portion of the operating system that is not associated with the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, compressing an additional portion of the operating system that is not associated with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1.1 above.	

9.2 storing the additional portion of the operating system in the first memory, and	Ballard, as evidenced by the example citations below, discloses "storing the additional portion of the operating system in the first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing the additional portion of the operating system in the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 8.1 above.	

9.3 wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system.	Ballard, as evidenced by the example citations below, discloses "wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 8.5 above.	

10. The method of claim 9, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder.	Ballard, as evidenced by the example citations below, discloses "the method of claim 9, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 9.1 above.	

Appendix C3
Invalidity of U.S. Patent 8,880,862 based on Ballard

11 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	Ballard, as evidenced by the example citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, comprising:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1 (Preamble) above.	

11.1 loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system;	Ballard, as evidenced by the example citations below, discloses "loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 1.1 and 6 (Preamble) above.	

11.2 accessing the loaded boot data in compressed form from the memory;	Ballard, as evidenced by the example citations below, discloses "accessing the loaded boot data in compressed form from the memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded boot data in compressed form from the memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1.2 above.	

11.3 decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form;	Ballard, as evidenced by the example citations below, discloses "decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1.3 above.	

11.4 utilizing the decompressed boot data to load at least a portion of the operating system for the computer system; and	Ballard, as evidenced by the example citations below, discloses "utilizing the decompressed boot data to load at least a portion of the operating system for the computer system"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed boot data to load at least a portion of the operating system for the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

> Launch time for a computer program is reduced by logging hard disk accesses during an initial launch, then processing the log file to accelerate subsequent launches. The log file is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. The disk access log entries are sorted by physical address or are grouped by file, then organized by logical address within each group. The processed log file is stored with the application program. When the application program is launched thereafter, the processed log file is accessed first. All the disk accesses in the log file are performed moving all the data into RAM cache. When the program launch resumes, the launch occurs faster because all the data is already in cache.

Ballard, Abstract

> It is common for an application program for a personal computer to be stored in multiple files on the secondary storage device. There often is an executable file, a preferences file and many other files. Some programs include a data base file or a default data file. During a launch of the program multiple files are opened and select portions are moved from the secondary storage device into the primary storage device. When purchasing a computer program the packaging often specifies the amount of RAM address space (i.e., primary storage device address space) required to be allocated to the program while active. By active it is meant that the program has been launched and is currently processing or is currently able to accept input commands.

Ballard, 1:51-63

According to another aspect of the invention, the launch access log is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. According to one approach the disk access log entries are sorted by physical address. According to another approach the disk access log entries are grouped by file, then organized by logical address within each group. The processed log file then is stored with the application program.

Ballard, 2:22-30

The primary storage device 14 typically is a storage device having a faster access time than that of the secondary storage device. An exemplary primary storage device 14 is random access memory (RAM). All or a portion of the RAM serves as a RAM cache 15. Portions of a computer program and/or data files are loaded into the RAM cache 15 to speed up execution of the program and processing of data. Mass produced computer software typically include specifications requiring a minimum amount of RAM required to run the program on a given computer system. During a launch sequence for starting such a computer program, portions of the program are copied from the secondary storage device into RAM.

A launch sequence as used herein means the sequence of steps executed by the computer system during the launch time which pertain to starting up a given computer program and getting the computer ready to accept input commands for the computer program. A launch sequence is executed for a computer program. Different computer programs have different launch sequences. Steps included in a launch sequence include copying portions of the computer program being launched from the secondary storage device to the primary storage device. Other steps may include allocating a port or device to serve as an input source and/or output receptor. The method of this invention for accelerating a program launch is directed to improving the speed for accessing the secondary storage device during a launch sequence.

Ballard, 3:40-67

Following is a description of the redundancy testing. Consider the following access sequence: address 139, address 119, address 110, addresses 120-133, and addresses 138-140. Also consider that the memory block and cluster size is 10 addresses, and that blocks are located at 100-109, 110-119, 120-129, 130-139, 140-149. When address 139 is accessed the contents within the block of addresses 130-139 is accessed. For an embodiment which stores a starting address of the cluster as the cluster address, the log entry includes address 130. When address 119 is accessed the contents within the block of addresses 110-

> 119 is accessed. The log entry for such access includes address 110. The next access in the launch sequence specifies address 110, causing the block 110-119 to be accessed. The cluster address is 110 which is already in the log. This access is a redundant access to a cluster already specified. The redundant access is eliminated by deleting the log entry for address 110. The next access specifies addresses 120-133. This access spans two clusters 120-129 and 130-139. The accesses are logged separately. The log entry for cluster address 120 is not redundant. The log entry for cluster address 130, however, is redundant because the first log entry for address 139 encompasses the memory block of addresses 130-139. To eliminate the redundancy, the log entry for address 139 is removed. The next access specifies addresses 139-140. This access also spans two clusters with two log entries. The first of the two is for cluster 130-139. This is a redundant entry and thus is removed from the log fie. The second of the two entries is for cluster 140-149. This is a nonredundant access. When the end of the log file is reached then redundancy testing (step 94) is complete.
>
> In many instances the redundant accesses are eliminated from consideration by a cache operation performed by the computer instead of by step 94. Specifically, when caching is performed the second access to the same cluster will not result in a call to the hard drive because the data is in the cache. The cache will satisfy the request. Requests satisfied by the cache are ignored for purposes of creating a log of accesses. Thus, redundant entries do not get logged.

Ballard, 7:15-53.

11.5 updating the boot data list.	Ballard, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1.4.1 above.	

13 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	Ballard, as evidenced by the example citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, comprising"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1 (Preamble) above.	

13.1 loading boot data in a compressed form that is associated with a boot data list from a boot device;	Ballard, as evidenced by the example citations below, discloses "loading boot data in a compressed form that is associated with a boot data list from a boot device"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading boot data in a compressed form that is associated with a boot data list from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1.1 above.	

13.2 accessing the loaded boot data in the compressed form;	Ballard, as evidenced by the example citations below, discloses "accessing the loaded boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1.2 above.	

13.3 decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form;	Ballard, as evidenced by the example citations below, discloses "decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1.3 above.	

13.4 updating the boot data list.	Ballard, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1.4.1 above.	

14 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	Ballard, as evidenced by the example citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, comprising"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1 (Preamble) above.	

14.1 accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list;	Ballard, as evidenced by the example citations below, discloses "accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

See Claims 1.1 and 1.2 above.

See also

> The log file corresponds to a specific computer program--the one being launched that triggered such log file to be created. When multiple programs are being launched at the same time, a log file is created for each such program. The interrupt service routine then sets a flag at step 78 to indicate that logging is enabled for such program. At step 80 the program returns. If the trigger for calling the routine 70 was for a secondary storage device access, then the steps at FIG. 4 also are performed before returning.
>
> Log File System Activity
>
> Once the launch of a computer program is detected and a log file is opened, all file system activity is monitored. Specifically, for each operating system call to the file system the call is analyzed to determine to which application being launched, if any, does the call pertain. Exemplary operating system calls to the file system are OPEN, READ, WRITE, and CLOSE. Referring to FIG. 4 routine 82 is entered at step 84 when both file system activity is detected and logging is enabled. If the call pertains to a computer program being launched, then an entry is appended to the appropriate log file (step 86). The routine 82 then returns at step 88.
>
> The log entry includes a file identifier, the logical address(es) specified in the call and the time of access (e.g., system time; index value).

Alternatively, the physical memory address(es) corresponding to the logical address(es) are stored in the log entry. In some embodiments, the operating system has already caused the physical addresses to be generated. If not, then the physical addresses are derived from the logical addresses using the operating system's file allocation table to translate the logical address into the physical address.

A logical address (also referred to as a virtual address) is the address which the computer program uses to access memory. A memory management unit translates this address into a physical address before the actual memory is read or written. A physical address is a memory location on the secondary storage device 16.

Ballard, 5:1-37

Detect Launch Completion

When completion of a launch sequence occurs and logging is enabled, then routine 90 is executed. Referring to FIG. 5, the routine enters at step 92. Conventional operating systems have a specific function that is called when a program is ready for normal execution. Under the Macintosh operating system, the function "Get Next Event" is called. For a computer program running under such operating system, such function is called by the computer program when the launch sequence is complete. According to one embodiment of this invention, step 92 is implemented by an interrupt service routine which is called whenever a computer program calls such function. The interrupt service routine clears the logging enabled flag for the corresponding application program.

In an alternative embodiment, access activity to the secondary storage device 16 is monitored to determine when activity has ceased for a threshold length of time (e.g., 3 seconds). Alternatively or in addition activity is monitored to determine when activity has gone below a threshold data throughput rate (e.g., 50 kilobytes per second) for a threshold period of time (e.g., 5 seconds). When there is insufficient activity for such threshold time, then the program launch is considered to be complete. The routine 90 then is executed.

Modify and Sort Log File

Once the launch sequence is determined to have been completed, then the log entry order is re-organized. The purpose is to eliminate redundant accesses to the same memory block and to optimize access time for the secondary storage device. If accesses occur faster, then the launch time (i.e., time elapsed from start to finish of launch sequence is less) is

> reduced. Thus, the launch of the computer program is accelerated.
>
> Referring to FIG. 5, at step 94 the log file is processed to eliminate log entries or log entry portions to redundant memory blocks.

Ballard, 5:55-6:23

> Note in this example that the log includes redundant entries. Entry h2 is a subset of entry d. Entry i2 is the same as entry f. Entry l2 is the same as entry a. It is expected that a redundant access request results from a computer program launch sequence access specifying a different address in the same block as previously accessed.
>
> Frequently the log file will include an entry specifying a single address. Even though only one address is specified, an entire memory block will be accessed (read or written). This is because the minimum file allocation unit is a memory block cluster. Thus, the smallest portion of the computer program than can be accessed is the cluster size (i.e., cluster size). Entries in the log file are tested at step 94 to identify any redundant accesses to portions of the computer program. To determine whether a later entry is an access to a redundant portion of the computer program, the cluster address for the access is identified. The cluster address is either stored in the log file or is derived from the address stored in the log file. For a FAT drive the cluster size is stored in the drive's boot sector. The boot sector includes information about the layout of the file system used for the drive partition. Following the boot sector are several reserved sectors. Following the reserved sectors is the file allocation table (FAT). Following the FAT are one or more backup copies of the FAT. Following the backup copies is the root directory. The size of the root directory is specified in the boot sector. After the root directory are the user's file and directory area. This is the area divided into clusters. Thus, from the information in the boot sector the starting address of cluster space is determined, along with the size of a cluster. Thus, the address boundaries for each cluster are known.
>
> The address for any given cluster x is given by Ax+B, where A and B are constants determined from the boot partition. Thus, for any given file system call the cluster boundaries for such address are determinable. The log file stores either the accessed address, the cluster number (i.e., x) or the cluster address (e.g., start address, end address or some other identifying address) for each cluster accessed.

Ballard, 6:44-7:14

> Following is a description of the redundancy testing. Consider the

> following access sequence: address 139, address 119, address 110, addresses 120-133, and addresses 138-140. Also consider that the memory block and cluster size is 10 addresses, and that blocks are located at 100-109, 110-119, 120-129, 130-139, 140-149. When address 139 is accessed the contents within the block of addresses 130-139 is accessed. For an embodiment which stores a starting address of the cluster as the cluster address, the log entry includes address 130. When address 119 is accessed the contents within the block of addresses 110-119 is accessed. The log entry for such access includes address 110. The next access in the launch sequence specifies address 110, causing the block 110-119 to be accessed. The cluster address is 110 which is already in the log. This access is a redundant access to a cluster already specified. The redundant access is eliminated by deleting the log entry for address 110. The next access specifies addresses 120-133. This access spans two clusters 120-129 and 130-139. The accesses are logged separately. The log entry for cluster address 120 is not redundant. The log entry for cluster address 130, however, is redundant because the first log entry for address 139 encompasses the memory block of addresses 130-139. To eliminate the redundancy, the log entry for address 139 is removed. The next access specifies addresses 139-140. This access also spans two clusters with two log entries. The first of the two is for cluster 130-139. This is a redundant entry and thus is removed from the log fie. The second of the two entries is for cluster 140-149. This is a nonredundant access. When the end of the log file is reached then redundancy testing (step 94) is complete.
>
> In many instances the redundant accesses are eliminated from consideration by a cache operation performed by the computer instead of by step 94. Specifically, when caching is performed the second access to the same cluster will not result in a call to the hard drive because the data is in the cache. The cache will satisfy the request. Requests satisfied by the cache are ignored for purposes of creating a log of accesses. Thus, redundant entries do not get logged.

Ballard, 7:15-53.

See also Ballard, Abstract, Tables B-C, Figs. 4-6.

14.2 loading the boot data into a memory; and	Ballard, as evidenced by the example citations below, discloses "loading the boot data into a memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the boot data into a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1.1 above.	

14.3 servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form; and	Ballard, as evidenced by the example citations below, discloses "servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

See Claims 1.2 and 1.3 above.

See also

> Launch time for a computer program is reduced by logging hard disk accesses during an initial launch, then processing the log file to accelerate subsequent launches. The log file is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. The disk access log entries are sorted by physical address or are grouped by file, then organized by logical address within each group. The processed log file is stored with the application program. When the application program is launched thereafter, the processed log file is accessed first. All the disk accesses in the log file are performed moving all the data into RAM cache. When the program launch resumes, the launch occurs faster because all the data is already in cache.

Ballard, Abstract

> A general purpose personal computer typically has many interactive application computer programs installed. A user is able to start-up

> multiple programs. With regard to an interactive computer program, the term "launch time", as used herein, means the time from when a processor receives a command to start the computer program until the time that the computer program is ready to accept input commands (e.g., user interface commands, batch-entry commands). The term "launch" as used herein means the process performed during the launch time to start up the computer program and get the computer ready to accept input commands for the computer progra[m].

Ballard, 1:38-50

> According to another aspect of the invention, the launch access log is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. According to one approach the disk access log entries are sorted by physical address. According to another approach the disk access log entries are grouped by file, then organized by logical address within each group. The processed log file then is stored with the application program.

Ballard, 2:22-30

> The primary storage device 14 typically is a storage device having a faster access time than that of the secondary storage device. An exemplary primary storage device 14 is random access memory (RAM). All or a portion of the RAM serves as a RAM cache 15. Portions of a computer program and/or data files are loaded into the RAM cache 15 to speed up execution of the program and processing of data. Mass produced computer software typically include specifications requiring a minimum amount of RAM required to run the program on a given computer system. During a launch sequence for starting such a computer program, portions of the program are copied from the secondary storage device into RAM.
>
> A launch sequence as used herein means the sequence of steps executed by the computer system during the launch time which pertain to starting up a given computer program and getting the computer ready to accept input commands for the computer program. A launch sequence is executed for a computer program. Different computer programs have different launch sequences. Steps included in a launch sequence include copying portions of the computer program being launched from the secondary storage device to the primary storage device. Other steps may include allocating a port or device to serve as an input source and/or output receptor. The method of this invention for accelerating a program launch is directed to improving the speed for accessing the secondary

> storage device during a launch sequence.

Ballard, 3:40-67

> Following is a description of the redundancy testing. Consider the following access sequence: address 139, address 119, address 110, addresses 120-133, and addresses 138-140. Also consider that the memory block and cluster size is 10 addresses, and that blocks are located at 100-109, 110-119, 120-129, 130-139, 140-149. When address 139 is accessed the contents within the block of addresses 130-139 is accessed. For an embodiment which stores a starting address of the cluster as the cluster address, the log entry includes address 130. When address 119 is accessed the contents within the block of addresses 110-119 is accessed. The log entry for such access includes address 110. The next access in the launch sequence specifies address 110, causing the block 110-119 to be accessed. The cluster address is 110 which is already in the log. This access is a redundant access to a cluster already specified. The redundant access is eliminated by deleting the log entry for address 110. The next access specifies addresses 120-133. This access spans two clusters 120-129 and 130-139. The accesses are logged separately. The log entry for cluster address 120 is not redundant. The log entry for cluster address 130, however, is redundant because the first log entry for address 139 encompasses the memory block of addresses 130-139. To eliminate the redundancy, the log entry for address 139 is removed. The next access specifies addresses 139-140. This access also spans two clusters with two log entries. The first of the two is for cluster 130-139. This is a redundant entry and thus is removed from the log fie. The second of the two entries is for cluster 140-149. This is a nonredundant access. When the end of the log file is reached then redundancy testing (step 94) is complete.
>
> In many instances the redundant accesses are eliminated from consideration by a cache operation performed by the computer instead of by step 94. Specifically, when caching is performed the second access to the same cluster will not result in a call to the hard drive because the data is in the cache. The cache will satisfy the request. Requests satisfied by the cache are ignored for purposes of creating a log of accesses. Thus, redundant entries do not get logged.

Ballard, 7:15-53.

14.4 updating the boot data list.	Ballard, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1.4.1 above.	

15. The method of claim 14, wherein the boot data comprises: a program code associated with the operating system.	Ballard, as evidenced by the example citations below, discloses "the method of claim 14, wherein the boot data comprises: a program code associated with the operating system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the boot data comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 1 (Preamble) and 1.1 above.	

Appendix C3
Invalidity of U.S. Patent 8,880,862 based on Ballard

16. The method of claim 14, wherein the operating system comprises: a plurality of files.	Ballard, as evidenced by the example citations below, discloses "the method of claim 14, wherein the operating system comprises: a plurality of files."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

> It is common for an application program for a personal computer to be stored in multiple files on the secondary storage device. There often is an executable file, a preferences file and many other files. Some programs include a data base file or a default data file. During a launch of the program multiple files are opened and select portions are moved from the secondary storage device into the primary storage device. When purchasing a computer program the packaging often specifies the amount of RAM address space (i.e., primary storage device address space) required to be allocated to the program while active. By active it is meant that the program has been launched and is currently processing or is currently able to accept input commands.

Ballard, 1:51-63

> According to another aspect of the invention, after program launch is complete the launch access log is processed. During a program's launch sequence multiple files are open at a given time. Contents from a first file are accessed, then contents from a second file. As the accesses continue the first file again is accessed. Thus, accesses to the multiple files are interspersed amongst each other. An exemplary sequence might be: File 1 (address 1), File 2 (address 20), File 3 (address 40), File 1 (address 6), File 3 (address 35), File 2 (address (25).

Ballard, 2:11-21

> FIG. 2 is a diagram of the secondary storage device 16 address space prior to a given computer program's launch sequence is accelerated. The dark regions indicate areas where contents 32-64 of a computer program 24 are stored. The device 16 also stores other computer programs and

> data. Address space allocation for such other programs and data is not shown. The computer program 24 includes multiple files. The minimum file system allocation unit is the smallest number of physical addresses that can be read or written to the secondary storage device. Such minimum file allocation unit also is referred to as a memory block or address cluster. When a specific address is specified in a READ call, the block encompassing such address is read from the secondary storage device and stored in the primary storage device. When a specific address is specified in a WRITE call, the block encompassing such address is written from the primary storage device into the secondary storage device. Listed below in table A is an exemplary portion of a file allocation table serving as a cross reference of logical addresses and physical addresses for the computer program (for a media having a cluster size of 10). Part numbers are added to correlate the table with FIG. 2. The logical addresses typically are generated at the time the computer program is compiled. The physical addresses are determined by the operating system when the computer program is installed. The physical addresses are the actual addresses on the secondary storage device 16 where the files are stored. Note that the part number is used to refer to a part of the computer program as distinct from the physical addresses at which such part is stored.

Ballard, 4:2-31; *see also* Ballard Fig. 2

17. The method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program.	Ballard, as evidenced by the example citations below, discloses "the method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

See Claim 15 above.

See also

> Launch time for a computer program is reduced by logging hard disk accesses during an initial launch, then processing the log file to accelerate subsequent launches. The log file is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. The disk access log entries are sorted by physical address or are grouped by file, then organized by logical address within each group. The processed log file is stored with the application program. When the application program is launched thereafter, the processed log file is accessed first. All the disk accesses in the log file are performed moving all the data into RAM cache. When the program launch resumes, the launch occurs faster because all the data is already in cache.

Ballard, Abstract

> A general purpose personal computer typically has many interactive application computer programs installed. A user is able to start-up multiple programs. With regard to an interactive computer program, the term "launch time", as used herein, means the time from when a processor receives a command to start the computer program until the time that the computer program is ready to accept input commands (e.g., user interface commands, batch-entry commands). The term "launch" as used herein means the process performed during the launch time to start up the computer program and get the computer ready to accept input commands

for the computer program.

It is common for an application program for a personal computer to be stored in multiple files on the secondary storage device. There often is an executable file, a preferences file and many other files. Some programs include a data base file or a default data file. During a launch of the program multiple files are opened and select portions are moved from the secondary storage device into the primary storage device. When purchasing a computer program the packaging often specifies the amount of RAM address space (i.e., primary storage device address space) required to be allocated to the program while active. By active it is meant that the program has been launched and is currently processing or is currently able to accept input commands.

Ballard, 1:38-63

The processor 12 serves to execute an operating system and one or more application computer programs. In some embodiments there are multiple processors for executing the operating system and application programs. System utilities and/or operating system extension programs also are executed according to some computer system 10 embodiments. Conventional operating systems include DOS, Windows, Windows NT, MacOS, OS/2 and various UNIX-based operating systems. The display device 18 and input devices 20 enable interaction between a user and the computer system 10. The computer system 10 in the process of executing the operating system and zero or more computer programs defines an operating environment for a user to interact with the computer, operating system and executing computer program. The display device 18 serves as an output device. Exemplary display devices include a CRT monitor or flat panel display. The user inputs commands and data to the computer system 10 via the input devices. Exemplary input devices include a keyboard, a pointing device and a clicking device. Data also is input to the computer via transportable disks or through I/O ports (not shown).

Ballard, 3:10-30.

Log File System Activity

Once the launch of a computer program is detected and a log file is opened, all file system activity is monitored. Specifically, for each operating system call to the file system the call is analyzed to determine to which application being launched, if any, does the call pertain. Exemplary operating system calls to the file system are OPEN, READ, WRITE, and CLOSE. Referring to FIG. 4 routine 82 is entered at step 84 when both file system activity is detected and logging is enabled. If the

> call pertains to a computer program being launched, then an entry is appended to the appropriate log file (step 86). The routine 82 then returns at step 88.
>
> The log entry includes a file identifier, the logical address(es) specified in the call and the time of access (e.g., system time; index value). Alternatively, the physical memory address(es) corresponding to the logical address(es) are stored in the log entry. In some embodiments, the operating system has already caused the physical addresses to be generated. If not, then the physical addresses are derived from the logical addresses using the operating system's file allocation table to translate the logical address into the physical address.
>
> A logical address (also referred to as a virtual address) is the address which the computer program uses to access memory. A memory management unit translates this address into a physical address before the actual memory is read or written. A physical address is a memory location on the secondary storage device 16.

Ballard, 5:1-37

19.1 The method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:	Ballard, as evidenced by the example citations below, discloses "the method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 1.1 and 1.2 above	

19.2 associating the accessed boot data that is not associated with the boot data list to the boot data list.	Ballard, as evidenced by the example citations below, discloses "associating the accessed boot data that is not associated with the boot data list to the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, associating the accessed boot data that is not associated with the boot data list to the boot data list.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 2 above.	

23. The method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files.	Ballard, as evidenced by the example citations below, discloses "the method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 16 above.	

24. The method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system.	Ballard, as evidenced by the example citations below, discloses "the method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system.
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 15 above.	

27. The method of claim 1, wherein the memory comprises: a physical memory.	Ballard, as evidenced by the example citations below, discloses "the method of claim 1, wherein the memory comprises: a physical memory."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

See Claims 1.1 and 1.2 above

See also

> The secondary storage device 16 serves as a permanent storage memory for one or more computer programs 24 to be executed by the processor 12. The secondary storage device 16 also stores data files for use with the application computer programs. Exemplary secondary storage devices include a hard disk drive, floppy disk drive, CD-ROM drive, bernoulli disk drive or other drive system for accessing permanent or replaceable disks, such as floppy disks, magnetic disks, magneto-optical disks, or optical disks.
>
> The primary storage device 14 typically is a storage device having a faster access time than that of the secondary storage device. An exemplary primary storage device 14 is random access memory (RAM). All or a portion of the RAM serves as a RAM cache 15. Portions of a computer program and/or data files are loaded into the RAM cache 15 to speed up execution of the program and processing of data. Mass produced computer software typically include specifications requiring a minimum amount of RAM required to run the program on a given computer system. During a launch sequence for starting such a computer program, portions of the program are copied from the secondary storage device into RAM.

Ballard, 3:31-52

28. The method of claim 1, wherein the operating system comprises: a plurality of files.	Ballard, as evidenced by the example citations below, discloses “the method of claim 1, wherein the operating system comprises: a plurality of files”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Ballard discloses this limitation: *See* Claim 16 above.	

29. The method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program.	Ballard, as evidenced by the example citations below, discloses "the method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 15 and 17 above.	

31. The method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access.	Ballard, as evidenced by the example citations below, discloses “the method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access”

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.

Ballard discloses this limitation:

See Claims 1.1 and 1.2 above

See also

> The secondary storage device 16 serves as a permanent storage memory for one or more computer programs 24 to be executed by the processor 12. The secondary storage device 16 also stores data files for use with the application computer programs. Exemplary secondary storage devices include a hard disk drive, floppy disk drive, CD-ROM drive, bernoulli disk drive or other drive system for accessing permanent or replaceable disks, such as floppy disks, magnetic disks, magneto-optical disks, or optical disks.
>
> The primary storage device 14 typically is a storage device having a faster access time than that of the secondary storage device. An exemplary primary storage device 14 is random access memory (RAM). All or a portion of the RAM serves as a RAM cache 15. Portions of a computer program and/or data files are loaded into the RAM cache 15 to speed up execution of the program and processing of data. Mass produced computer software typically include specifications requiring a minimum amount of RAM required to run the program on a given computer system. During a launch sequence for starting such a computer program, portions of the program are copied from the secondary storage device into RAM.

Ballard, 3:31-52

32. The method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form.	Ballard, as evidenced by the example citations below, discloses "the method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 1.1 above	

33. The method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form.	Ballard, as evidenced by the example citations below, discloses "the method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 1.1 and 32 above.	

35. The method of claim 5, wherein the compressed boot data represents a plurality of files.	Ballard, as evidenced by the example citations below, discloses "the method of claim 5, wherein the compressed boot data represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the compressed boot data represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 16 and 23 above.	

36. The method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system.	Ballard, as evidenced by the example citations below, discloses "the method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 15 and 17 above.	

39. The method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory.	Ballard, as evidenced by the example citations below, discloses "the method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

See Claims 1.1 and 1.2 above

See also

> The secondary storage device 16 serves as a permanent storage memory for one or more computer programs 24 to be executed by the processor 12. The secondary storage device 16 also stores data files for use with the application computer programs. Exemplary secondary storage devices include a hard disk drive, floppy disk drive, CD-ROM drive, bernoulli disk drive or other drive system for accessing permanent or replaceable disks, such as floppy disks, magnetic disks, magneto-optical disks, or optical disks.
>
> The primary storage device 14 typically is a storage device having a faster access time than that of the secondary storage device. An exemplary primary storage device 14 is random access memory (RAM). All or a portion of the RAM serves as a RAM cache 15. Portions of a computer program and/or data files are loaded into the RAM cache 15 to speed up execution of the program and processing of data. Mass produced computer software typically include specifications requiring a minimum amount of RAM required to run the program on a given computer system. During a launch sequence for starting such a computer program, portions of the program are copied from the secondary storage device into RAM.

Ballard, 3:31-52

Appendix C3
Invalidity of U.S. Patent 8,880,862 based on Ballard

40. The method of claim 36, wherein the operating system comprises: a plurality of files.	Ballard, as evidenced by the example citations below, discloses "the method of claim 36, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 36, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 16 and 23 above.	

43. The method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access.	Ballard, as evidenced by the example citations below, discloses "the method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 31 above.	

44. The method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data.	Ballard, as evidenced by the example citations below, discloses “the method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data”

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.

Ballard discloses this limitation:

See Claim 32 above.

45. The method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data.	Ballard, as evidenced by the example citations below, discloses "the method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 32 and 33 above.	

47. The system of claim 6, wherein the boot data in the compressed form represents a plurality of files.	Ballard, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 16 and 23 above.	

48. The system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system.	Ballard, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 15, 17 and 24 above.	

49. The system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form.	Ballard, as evidenced by the example citations below, discloses “the system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Ballard discloses this limitation: *See* Claim 10 above.	

50. The system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form.	Ballard, as evidenced by the example citations below, discloses "the system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

> Launch time for a computer program is reduced by logging hard disk accesses during an initial launch, then processing the log file to accelerate subsequent launches. The log file is processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. The disk access log entries are sorted by physical address or are grouped by file, then organized by logical address within each group. The processed log file is stored with the application program. When the application program is launched thereafter, the processed log file is accessed first. All the disk accesses in the log file are performed moving all the data into RAM cache. When the program launch resumes, the launch occurs faster because all the data is already in cache.

Ballard, Abstract

> A general purpose personal computer typically has many interactive application computer programs installed. A user is able to start-up multiple programs. With regard to an interactive computer program, the term "launch time", as used herein, means the time from when a processor receives a command to start the computer program until the time that the computer program is ready to accept input commands (e.g., user interface commands, batch-entry commands). The term "launch" as used herein means the process performed during the launch time to start up the computer program and get the computer ready to accept input commands for the computer progra[m].

Ballard, 1:38-50

> According to another aspect of the invention, the launch access log is

processed by identifying all the file portions accessed during the launch, eliminating any duplicate cluster accesses, then sorting the remaining accesses. According to one approach the disk access log entries are sorted by physical address. According to another approach the disk access log entries are grouped by file, then organized by logical address within each group. The processed log file then is stored with the application program.

Ballard, 2:22-30

The primary storage device 14 typically is a storage device having a faster access time than that of the secondary storage device. An exemplary primary storage device 14 is random access memory (RAM). All or a portion of the RAM serves as a RAM cache 15. Portions of a computer program and/or data files are loaded into the RAM cache 15 to speed up execution of the program and processing of data. Mass produced computer software typically include specifications requiring a minimum amount of RAM required to run the program on a given computer system. During a launch sequence for starting such a computer program, portions of the program are copied from the secondary storage device into RAM.

A launch sequence as used herein means the sequence of steps executed by the computer system during the launch time which pertain to starting up a given computer program and getting the computer ready to accept input commands for the computer program. A launch sequence is executed for a computer program. Different computer programs have different launch sequences. Steps included in a launch sequence include copying portions of the computer program being launched from the secondary storage device to the primary storage device. Other steps may include allocating a port or device to serve as an input source and/or output receptor. The method of this invention for accelerating a program launch is directed to improving the speed for accessing the secondary storage device during a launch sequence.

Ballard, 3:40-67

51. The system of claim 6, wherein the first memory comprises: a physical memory.	Ballard, as evidenced by the example citations below, discloses "the system of claim 6, wherein the first memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the first memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 27 and 39 above.	

52. The system of claim 6, wherein the boot data the compressed form comprises: a plurality of files.	Ballard, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data the compressed form comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data the compressed form comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 16 and 23 above.	

53. The system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program.	Ballard, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 15, 17 and 24 above.	

Appendix C3
Invalidity of U.S. Patent 8,880,862 based on Ballard

59. The method of claim 8, wherein the operating system in the compressed form represents a plurality of files.	Ballard, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 16 and 23 above.	

60. The method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system.	Ballard, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 15, 17 and 24 above.	

63. The method of claim 8, wherein the second memory comprises: a physical memory.	Ballard, as evidenced by the example citations below, discloses "the method of claim 8, wherein the second memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the second memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 27 and 38 above.	

64. The method of claim 8, wherein the operating system comprises: a plurality of files.	Ballard, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 16 and 23 above.	

65. The method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program.	Ballard, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 15, 17 and 24 above.	

67. The method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access.	Ballard, as evidenced by the example citations below, discloses "the method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 31 above.	

71. The method of claim 11, wherein the boot data in the compressed form represents a plurality of files.	Ballard, as evidenced by the example citations below, discloses "the method of claim 11, wherein the boot data in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 16 and 23 above.	

72. The method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system.	Ballard, as evidenced by the example citations below, discloses "the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 15, 17 and 24 above.	

75. The method of claim 11, wherein the memory comprises: a physical memory.	Ballard, as evidenced by the example citations below, discloses "the method of claim 11, wherein the memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 27 above.	

76. The method of claim 11, wherein the operating system comprises: a plurality of files.	Ballard, as evidenced by the example citations below, discloses “the method of claim 11, wherein the operating system comprises: a plurality of files”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Ballard discloses this limitation: *See* Claims 16 and 23 above.	

77. The method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program.	Ballard, as evidenced by the example citations below, discloses "the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 15, 17 and 24 above.	

79. The method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access.	Ballard, as evidenced by the example citations below, discloses "the method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 31 above.	

80. The method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form.	Ballard, as evidenced by the example citations below, discloses "the method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 32, 33 and 49 above.	

81. The method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form.	Ballard, as evidenced by the example citations below, discloses "the method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 32, 33 and 49 above.	

83. The method of claim 13, wherein the boot data in the compressed form represents a plurality of files.	Ballard, as evidenced by the example citations below, discloses "the method of claim 13, wherein the boot data in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 16 and 23 above.	

84. The method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system.	Ballard, as evidenced by the example citations below, discloses "the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 15, 17 and 24 above.	

87. The method of claim 13, wherein the memory comprises: a physical memory.	Ballard, as evidenced by the example citations below, discloses "the method of claim 13, wherein the memory comprises: a physical memory"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

See Claim 27 above.

88. The method of claim 13, wherein the operating system comprises: a plurality of files.	Ballard, as evidenced by the example citations below, discloses "the method of claim 13, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 16 and 23 above.	

89. The method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program.	Ballard, as evidenced by the example citations below, discloses "the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 15, 17 and 24 above.	

91. The method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access.	Ballard, as evidenced by the example citations below, discloses "the method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 31 above.	

92. The method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data.	Ballard, as evidenced by the example citations below, discloses "the method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 32, 33 and 49 above.	

93. The method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data.	Ballard, as evidenced by the example citations below, discloses "the method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claim 32, 33, and 49 above.	

97.1 The method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list,	Ballard, as evidenced by the example citations below, discloses "the method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 9.1-9.3 and 19 above.	

97.2 and wherein the updating comprises: associating the additional compressed boot data with the boot data list.	Ballard, as evidenced by the example citations below, discloses "and wherein the updating comprises: associating the additional compressed boot data with the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, and wherein the updating comprises: associating the additional compressed boot data with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 1.4.1, 2, 4, 5.6, and 8.6 above.	

98. The method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list.	Ballard, as evidenced by the example citations below, discloses "the method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 1.4.1, 5.6, and 8.6 above.	

107. The method of claim 2, further comprising: storing the updated boot list in a non-volatile memory.	Ballard, as evidenced by the example citations below, discloses "the method of claim 2, further comprising: storing the updated boot list in a non-volatile memory."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 2, further comprising: storing the updated boot list in a non-volatile memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ballard discloses this limitation:

See Claims 1.4.1, 5.6, and 8.6 above.

See also

> The secondary storage device 16 serves as a permanent storage memory for one or more computer programs 24 to be executed by the processor 12. The secondary storage device 16 also stores data files for use with the application computer programs. Exemplary secondary storage devices include a hard disk drive, floppy disk drive, CD-ROM drive, bernoulli disk drive or other drive system for accessing permanent or replaceable disks, such as floppy disks, magnetic disks, magneto-optical disks, or optical disks.
>
> The primary storage device 14 typically is a storage device having a faster access time than that of the secondary storage device. An exemplary primary storage device 14 is random access memory (RAM). All or a portion of the RAM serves as a RAM cache 15. Portions of a computer program and/or data files are loaded into the RAM cache 15 to speed up execution of the program and processing of data. Mass produced computer software typically include specifications requiring a minimum amount of RAM required to run the program on a given computer system. During a launch sequence for starting such a computer program, portions of the program are copied from the secondary storage device into RAM.

Ballard, 3:31-52

108. The method of claim 2, further comprising: compressing at least a portion of the additional boot data.	Ballard, as evidenced by the example citations below, discloses "the method of claim 2, further comprising: compressing at least a portion of the additional boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 2, further comprising: compressing at least a portion of the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 1.1, 5, 9.1 and 8 above.	

109. The method of claim 108, further comprising: storing the compressed additional boot data.	Ballard, as evidenced by the example citations below, discloses "the method of claim 108, further comprising: storing the compressed additional boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 108, further comprising: storing the compressed additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ballard discloses this limitation: *See* Claims 5.1, 8.1 and 9.1 above.	

Appendix C4
Invalidity of U.S. Patent 8,880,862 based on Feigenbaum

U.S. Patent No. 5,307,497 to Feigenbaum ("Feigenbaum") invalidates claims 1-6, 8-11, 13-17, 19, 23-24, 27-29, 31-33, 35-36, 39-40, 43-45, 47-53, 59-60, 63-65, 67, 71-72, 75-77, 79-81, 83-84, 87-89, 91-93, 97-98, and 107-109 of United States Patent No. 8,880,862 ("the '862 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '862 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix C4
Invalidity of U.S. Patent 8,880,862 based on Feigenbaum

1 (Preamble) A method for providing accelerated loading of an operating system in a computer system, the method comprising:	Feigenbaum, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, the method comprising:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system in a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

> A data processing system, such as a personal computer, contains bootable DOS programs that are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation.

Feigenbaum, Abstract

> This invention relates to the field of data processing, and, more particularly, to a method and apparatus for loading a personal computer disk operating system (DOS) from a read only memory (ROM) using an installable file system (IFS) interface.

Feigenbaum, 1:7-12

> When such personal computers are powered up, a power on self test (POST) routine or program is first executed, such program being stored in ROM. Upon the successful completion of such test, portions of DOS are read into the system memory from a hard disk or a floppy diskette and the system is booted up and initialized. The test, booting and initialization process can take many seconds. The present invention is directed to an improvement by which such process is significantly shortened in time. It is also common to start a personal computer by turning on not only a system unit but also a display, and the invention has an objective of accomplishing the start up process so that the system is available for use within the time required to warm up the display. That is, as soon as the system is turned on, the first screen becomes immediately visible on a display and gives the user the appearance of an "instant" load and initialization.

Feigenbaum, 1:22-39

> It is also known that the speed at which a ROM can be accessed is several orders of magnitude faster than the speed at which a hard disk or a floppy

> diskette can be accessed. The invention takes advantage of this known speed difference by storing portions of DOS in a ROM where such portions can be accessed at a speed much faster than the speed at which DOS could be accessed if such portions of DOS were stored on a hard disk or floppy diskette. However, the invention is more than simply substituting a ROM for a disk because of several problems. A first thought that might occur to many persons is that by storing DOS in a ROM, DOS can then be executed directly from the ROM. However, while certain portions of DOS, such as commands in the COMMAND.COM program, can be executed directly from ROM, other portions, such as IBMBIO.COM and IBMDOS.COM are altered and cannot be used in a read only device which precludes any alteration.

Feigenbaum, 1:40-57

> One of the objects of the invention is to provide a ROM based DOS that rapidly tests, boots, and initializes a personal computer and appears to the user to "instantly" start up when the computer is turned on.

Feigenbaum, 3:30-33

> A further object is to provide a ROM based DOS which uses an IFS that is accessible at the system level, as opposed to the device level, to rapidly load DOS programs from ROM into RAM for execution from such RAM.

Feigenbaum, 3:38-42

> Briefly, in accordance with the subject invention, bootable DOS programs are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation.

Feigenbaum, 3:53-59

> When switch 35 is initially turned on, display 30 warms up within a relatively short period and the ROM based booting and system initialization, described in detail below, occurs within the period required for the display to warm up so that at the end of such period, the display screen becomes visible to the user to give the appearance of an instant startup.

Feigenbaum, 4:63-5:2

> In summary, the invention of a ROM based DOS provides several features and advantages. DOS is rapidly loaded from ROM and executed in RAM to instantly boot up and present the user with a screen from a graphical user interface. Once booted, ROM DOS executes normally and acts the same as

> standard disk DOS and has the same system compatibility as disk DOS. Further, no DOS installation is required to be done by a user, and the user has no diskette dependencies. The invention is further advantageous by virtue of using the IFS interface and allowing the ROM to appear as a "drive", since this minimizes the number of changes to the basic DOS programs, which reduced developement time and minimized the possibility of error being introduced by new code.

Feigenbaum, 10:18-33

Appendix C4
Invalidity of U.S. Patent 8,880,862 based on Feigenbaum

1.1 loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory;	Feigenbaum, as evidenced by the example citations below, discloses "loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

> A data processing system, such as a personal computer, contains bootable DOS programs that are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation.

Feigenbaum, Abstract

> Another solution that might occur would be to create the ROM within the DOS address space, or within the first one megabyte of memory mapped space, and then load or copy DOS into a RAM (random access memory) within such address space to occupy and execute from the same space in RAM and operate in the same manner as DOS currently does. The disadvantage of such a solution is that two copies of DOS would then exist in such limited address space, one copy being the unalterable ROM DOS and the other copy being the alterable one that is stored in and executed from the RAM. Having two copies of essentially the same program in such a limited address space would not be efficient use of memory nor acceptable by many users.

Feigenbaum, 1:58-2:3

> Briefly, in accordance with the subject invention, bootable DOS programs are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation.

Feigenbaum, 1:53-60

> Referring to FIG. 3, ROM DOS 34 includes various files for loading the DOS kernel and starting system 10 with a shell program. The first 512 bytes comprise ROM identification information 42, a jump instruction 44, module information and data 46, and a ROM BOOT program 48. ROM DOS 34 further includes an IBMBIO.COM program 50 including a RAM LOADER 51, an IBMDOS.COM program 52 including an IFS handler 53, a resident COMMAND program 54, a transient COMMAND program 57, a SHELSTUB program 59, a ROMSHELL program 57, a mouse driver MOUSE.COM 60, an AUTOEXEC.BAT file 61, a CONFIG.SYS file 62, and a CHECKC.COM program 63. Items 42 through 48 are stored in the first 512 bytes of ROM DOS 34 and are comparable to the conventional bootstrap record stored in the first sector of disk 24. The final steps performed by POST 40 are to copy the first 512 bytes of ROM DOS 34 into RAM 14 and then to transfer control, by a jump instruction, to jump instruction 44 which in turn jumps to the start of ROM BOOT program 48.

Feigenbaum, 5:50-6:2

> Programs 50 through 56 provide the minimum level of operating system support for operating data processing system 10. IBMBIO.COM 50 (except for RAM LOADER 51), IBMDOS.COM 52 (except for IFS handler 53), and COMMANDS 54 AND 56 (which together form the conventional COMMAND.COM DOS program) comprise the conventional DOS kernel. IBMBIO.COM provides low level device support. IBMDOS.COM provides application support. COMMAND.COM is the command processor. COMMAND 54 stays resident in RAM 14 once it is loaded while COMMAND 56 can be overlaid and later copied into RAM 14 as needed. ROMSHELL 57 provides a graphical user interface and is transient while SHELSTUB 59 is designed to be resident in the RAM for reloading the ROMSHELL program after quitting an application. The ROM DOS 34 programs when loaded into the RAM generally operate the same as the corresponding programs in DISK DOS 36, the only difference being that such programs are loaded from ROM 16 instead of disk 24. Relative to other portions of DOS and the system, ROM DOS 34 is transparent and appears to act the same as corresponding portions of DISK DOS 36. The ROM DOS files are stored in ROM 16 outside of the address space available for DOS programs, and RAM LOADER 51 is able to work outside such address space to copy such programs into the DOS address space in RAM 14. RAM LOADER 51 uses a standard method of moving blocks of data from the protected mode address space into the real mode DOS address space, such as by using the well known system service interrupt 15H of BIOS 40 which is invoked by setting the well known register AH (not shown) of microprocessor 12 to the MOVE BLOCK function 87H. Such function transfers a block of data from storage above the IMB protected mode address range by switching to the protected mode. RAM LOADER 51 is invoked using interrupt 2BH and accesses the information from items 42 and 46 to load the

ROM header at initialization time, check for the existence of a module in ROM, load a module from ROM, and access flag 58 in CMOS RAM 20.

Feigenbaum, 6:3-44

RAM LOADER 51, IFS HANDLER 53, and CHECKC.COM 63 are new with the invention, while the remaining programs stored in ROM 16 perform functions previously commercially available. ROM 16 may optionally be of a larger size to store additional programs such as a code page switching program, a program providing extended keyboard layout, support for other national languages, etc. It may also contain alternative AUTOEXEC.BAT and CONFIG.SYS files 102 and 104 oriented to the use of such additional programs. The size of ROM 16 may thus vary dependent upon the size of and how many additional support programs are stored. Preferably, those items shown in FIG. 3 are stored in a 128K ROM unit and the additional programs are stored in a second 128 K ROM unit.

Feigenbaum, 6:45-59

ROMBOOT 65 first loads IBMBIO.COM 50, including RAM LOADER 51, from ROM 16 into RAM 14 by step 76. IBMBIO.COM includes two code segments, IBMINIT and SYSINIT. Step 78 then transfers control to IBMINIT 66. Step 80 initializes the system device drivers and hardware. Step 81 provides or sets up a hook into interrupt handler INT 2BH for the RAM LOADER. Step 82 relocates the initialization routine of IBMBIO.COM 50 to predetermined locations at the top of the DOS address space in RAM 16. Next, step 84 transfers control to SYSINIT 67 which first, by step 86, loads IBMDOS.COM 52 from ROM 16 into RAM 14. Step 88 then executes the initialization routine of IBMDOS.COM. Afterwards, step 90 attaches the IFS Handler as the next available drive which is drive D: because system 10 includes a fixed disk drive. Next, step 91 selects the ROM drive created by the IFS handler, as the defualt drive.

Feigenbaum, 8:53-9:2

In summary, the invention of a ROM based DOS provides several features and advantages. DOS is rapidly loaded from ROM and executed in RAM to instantly boot up and present the user with a screen from a graphical user interface. Once booted, ROM DOS executes normally and acts the same as standard disk DOS and has the same system compatibility as disk DOS. Further, no DOS installation is required to be done by a user, and the user has no diskette dependencies. The invention is further advantageous by virtue of using the IFS interface and allowing the ROM to appear as a "drive", since this minimizes the number of changes to the basic DOS programs, which reduced developement time and minimized the possibility of error being introduced by new code.

Feigenbaum, 10:18-33

1.2 accessing the loaded portion of the boot data in the compressed form from memory;	Feigenbaum, as evidenced by the example citations below, discloses "accessing the loaded portion of the boot data in the compressed form from memory;"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded portion of the boot data in the compressed form from memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: > Another object is to store a DOS kernel in a ROM in a simple, efficient file structure that can be rapidly accessed using less code than that required to access a DOS kernel stored on a disk in a FAT file system. Feigenbaum, 3:34-37 > A still further object is to provide improved method and apparatus for rapidly initializing a personal computer when it is turned on, which method and apparatus uses an IFS to access DOS programs stored in a ROM in a packed format differing from the format used to store such programs in a FAT file system. Feigenbaum, 3:47-52 > Programs 50 through 56 provide the minimum level of operating system support for operating data processing system 10. IBMBIO.COM 50 (except for RAM LOADER 51), IBMDOS.COM 52 (except for IFS handler 53), and COMMANDS 54 AND 56 (which together form the conventional COMMAND.COM DOS program) comprise the conventional DOS kernel. IBMBIO.COM provides low level device support. IBMDOS.COM provides application support. COMMAND.COM is the command processor. COMMAND 54 stays resident in RAM 14 once it is loaded while COMMAND 56 can be overlaid and later copied into RAM 14 as needed. ROMSHELL 57 provides a graphical user interface and is transient while SHELSTUB 59 is designed to be resident in the RAM for reloading the ROMSHELL program after quitting an application. The ROM DOS 34 programs when loaded into the RAM generally operate the same as the corresponding programs in DISK DOS 36, the only difference being that such programs are loaded from ROM 16 instead of disk 24. Relative to other portions of DOS and the system, ROM DOS 34 is transparent and appears to act the same as corresponding portions of DISK DOS	

36. The ROM DOS files are stored in ROM 16 outside of the address space available for DOS programs, and RAM LOADER 51 is able to work outside such address space to copy such programs into the DOS address space in RAM 14. RAM LOADER 51 uses a standard method of moving blocks of data from the protected mode address space into the real mode DOS address space, such as by using the well known system service interrupt 15H of BIOS 40 which is invoked by setting the well known register AH (not shown) of microprocessor 12 to the MOVE BLOCK function 87H. Such function transfers a block of data from storage above the IMB protected mode address range by switching to the protected mode. RAM LOADER 51 is invoked using interrupt 2BH and accesses the information from items 42 and 46 to load the ROM header at initialization time, check for the existence of a module in ROM, load a module from ROM, and access flag 58 in CMOS RAM 20.

Feigenbaum, 6:3-6:44

FIG. 4 illustrates the relative position and use of IFS HANDLER 53. Within the prior art, an application program 110 can be thought of as sitting on top of FAT file system 112, which in turn sits on top of disk/diskette driver 114 which overlies the disk 24 containing the actual files and data. Application 110 uses interrupt INT 21H to access the file system through IBMDOS.COM, which acts through IBMBIO.COM to access the disk. In accordance with the invention, IFS HANDLER 53 is interposed at the file system level in the INT 21H process and decides in step 116 whether the desired file system access is to be made to ROM DISK D:. If not, the system proceeds as in the prior art to access the FAT file system. If the desired access is to be made to ROM DISK D:, it is done through the ROM file system 118 and RAM LOADER 51 by programs IBMDOS.COM 52 and IBMBIO.COM 50. The term "ROM DISK" is defined hereby to mean a ROM unit containing images of files storable on a hard disk or floppy diskette, which images are stored in a packed format in a ROM file system, that is different from the FAT file system used to store files on a disk or diskette.

Feigenbaum, 9:39-60

The IFS interface "connects" the ROM as an installable file system and assigns it a drive letter to be referenced. Such interface performs calls at a file system level as opposed to other systems which operate at the disk level. Examples of interfaces operating at the system level are OPEN, READ, CLOSE, etc. Examples of interfaces which operate at the disk level are READ SECTOR, and WRITE SECTOR. Operating system performance is improved since file system calls are identified earlier in the code path than disk calls. Further, in view of the relative simplicity of the ROM file structure in comparison to the complex file structure of the FAT file system, it is obvious that the amount of code necessary to access the ROM file system is substantially less than the amount of code necessary to access the FAT file system. Thus performance is faster because less code has to be executed and because the FAT file system code is bypassed due

to the system level interrupt being located in the code path prior to the FAT file system code in IBMDOS.COM. Feigenbaum, 9:61-10:12

1.3 decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form; and	Feigenbaum, as evidenced by the example citations below, discloses "decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

> A data processing system, such as a personal computer, contains bootable DOS programs that are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation.

Feigenbaum, Abstract

> When such personal computers are powered up, a power on self test (POST) routine or program is first executed, such program being stored in ROM. Upon the successful completion of such test, portions of DOS are read into the system memory from a hard disk or a floppy diskette and the system is booted up and initialized. The test, booting and initialization process can take many seconds. The present invention is directed to an improvement by which such process is significantly shortened in time. It is also common to start a personal computer by turning on not only a system unit but also a display, and the invention has an objective of accomplishing the start up process so that the system is available for use within the time required to warm up the display. That is, as soon as the system is turned on, the first screen becomes immediately visible on a display and gives the user the appearance of an "instant" load and initialization.

Feigenbaum, 1:22-39

> It is also known that the speed at which a ROM can be accessed is several orders of magnitude faster than the speed at which a hard disk or a floppy diskette can be accessed. The invention takes advantage of this known speed difference by

> storing portions of DOS in a ROM where such portions can be accessed at a speed much faster than the speed at which DOS could be accessed if such portions of DOS were stored on a hard disk or floppy diskette. However, the invention is more than simply substituting a ROM for a disk because of several problems. A first thought that might occur to many persons is that by storing DOS in a ROM, DOS can then be executed directly from the ROM. However, while certain portions of DOS, such as commands in the COMMAND.COM program, can be executed directly from ROM, other portions, such as IBMBIO.COM and IBMDOS.COM are altered and cannot be used in a read only device which precludes any alteration.

Feigenbaum, 1:40-57

> In summary, the invention of a ROM based DOS provides several features and advantages. DOS is rapidly loaded from ROM and executed in RAM to instantly boot up and present the user with a screen from a graphical user interface. Once booted, ROM DOS executes normally and acts the same as standard disk DOS and has the same system compatibility as disk DOS. Further, no DOS installation is required to be done by a user, and the user has no diskette dependencies. The invention is further advantageous by virtue of using the IFS interface and allowing the ROM to appear as a "drive", since this minimizes the number of changes to the basic DOS programs, which reduced developement time and minimized the possibility of error being introduced by new code.

Feigenbaum, 10:18-33

1.4.1 updating the boot data list,	Feigenbaum, as evidenced by the example citations below, discloses "updating the boot data list,"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

> Another solution that might occur would be to create the ROM within the DOS address space, or within the first one megabyte of memory mapped space, and then load or copy DOS into a RAM (random access memory) within such address space to occupy and execute from the same space in RAM and operate in the same manner as DOS currently does. The disadvantage of such a solution is that two copies of DOS would then exist in such limited address space, one copy being the unalterable ROM DOS and the other copy being the alterable one that is stored in and executed from the RAM. Having two copies of essentially the same program in such a limited address space would not be efficient use of memory nor acceptable by many users.

Feigenbaum, 1:58-2:3

1.4.2 wherein the decompressed portion of boot data comprises a portion of the operating system.	Feigenbaum, as evidenced by the example citations below, discloses "wherein the decompressed portion of boot data comprises a portion of the operating system."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the decompressed portion of boot data comprises a portion of the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

> A data processing system, such as a personal computer, contains bootable DOS programs that are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation.

Feigenbaum, Abstract

> When such personal computers are powered up, a power on self test (POST) routine or program is first executed, such program being stored in ROM. Upon the successful completion of such test, portions of DOS are read into the system memory from a hard disk or a floppy diskette and the system is booted up and initialized. The test, booting and initialization process can take many seconds. The present invention is directed to an improvement by which such process is significantly shortened in time. It is also common to start a personal computer by turning on not only a system unit but also a display, and the invention has an objective of accomplishing the start up process so that the system is available for use within the time required to warm up the display. That is, as soon as the system is turned on, the first screen becomes immediately visible on a display and gives the user the appearance of an "instant" load and initialization.

Feigenbaum, 1:22-39

> It is also known that the speed at which a ROM can be accessed is several orders of magnitude faster than the speed at which a hard disk or a floppy diskette can be accessed. The invention takes advantage of this known speed difference by storing portions of DOS in a ROM where such portions can be accessed at a speed much faster than the speed at which DOS could be accessed if such portions of DOS were stored on a hard disk or floppy diskette. However, the invention is more than simply substituting a ROM for a disk because of several problems. A first thought that might occur to many persons is that by

> storing DOS in a ROM, DOS can then be executed directly from the ROM. However, while certain portions of DOS, such as commands in the COMMAND.COM program, can be executed directly from ROM, other portions, such as IBMBIO.COM and IBMDOS.COM are altered and cannot be used in a read only device which precludes any alteration.

Feigenbaum, 1:40-57

> System 10 is further provided with a DOS 32 that comprises two major portions, ROM DOS 34 and DISK DOS 36 respectively stored in ROM 16 and disk 24. ROM DOS 34 includes the programs that are booted into the RAM when the system is initially powered on or when the system is reset, which programs provide the minimum level of operating system support to make system 10 operational to the user. The remaining portions of DOS, i.e. those programs and files that together form a complete DOS, are included in DISK DOS 36. DISK DOS 36 further includes DOS programs similar to those in ROM DOS so that the system can be booted up and operated from disk 24 at the option of the user, as described in more detail hereafter. Also stored in ROM 16 are a basic input/output system (BIOS) 38 and a power on self test (POST) routine 40. When system 10 is first turned on, or when it is restarted, POST 40 first performs the test and upon successful completion, it boots or loads a portion of DOS and transfers control to it in the manner described in detail below. Except for ROM DOS 34, the system as thus far described is constructed and operates in accordance with known principles of the prior art.

Feigenbaum, 5:3-25

> In summary, the invention of a ROM based DOS provides several features and advantages. DOS is rapidly loaded from ROM and executed in RAM to instantly boot up and present the user with a screen from a graphical user interface. Once booted, ROM DOS executes normally and acts the same as standard disk DOS and has the same system compatibility as disk DOS. Further, no DOS installation is required to be done by a user, and the user has no diskette dependencies. The invention is further advantageous by virtue of using the IFS interface and allowing the ROM to appear as a "drive", since this minimizes the number of changes to the basic DOS programs, which reduced developement time and minimized the possibility of error being introduced by new code.

Feigenbaum, 10:18-33

Appendix C4
Invalidity of U.S. Patent 8,880,862 based on Feigenbaum

2. The method of claim 1, wherein the updating comprises: associating additional boot data with the boot data list.	Feigenbaum, as evidenced by the example citations below, discloses "wherein the updating comprises: associating additional boot data with the boot data list."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein associating additional boot data with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

See Claim 1.4.1 above.

> FIG. 3 also illustrates another facet of the invention which is that the data and programs of ROM DOS 34 are stored in ROM 16 in a file system using a packed format. In contrast, any data and files stored on disk 24, including those in RAM DOS 36, are stored in the conventional DOS FAT file system in which information is stored in clusters and sectors. In the FAT file system, if a particular program doesn't have the same number of bytes as are in a sector or cluster, space is wasted. On the average, about one half the number of bytes in a cluster or in a sector are wasted for each file. Since ROM units are more expensive than disk storage, any wasted space is inefficient. Thus, in ROM 16, the ROM DOS 34 files are stored beginning at a segment address, each segment being sixteen bytes. Each succeeding file begins immediately at the next segment location following the end of a preceding file so that on average only eight bytes per file would be wasted, such number being far less than the average waste in a FAT file system. By way of example, suppose file 50 begins at ROM segment X and ends at Y. Then, the succeeding file 52 begins at segment Y+n, where n is less than sixteen. The other files are similarly stored.

Feigenbaum, 6:60-7:15

3. The method of claim 1, wherein the updating comprises: removing an association of additional boot data that is associated with the boot data list from the boot data list.	Feigenbaum, as evidenced by the example citations below, discloses "wherein the updating comprises: removing an association of additional boot data that is associated with the boot data list from the boot data list."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein removing an association of additional boot data that is associated with the boot data list from the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

See Claim 1.4.1 above.

> It is also known that the speed at which a ROM can be accessed is several orders of magnitude faster than the speed at which a hard disk or a floppy diskette can be accessed. The invention takes advantage of this known speed difference by storing portions of DOS in a ROM where such portions can be accessed at a speed much faster than the speed at which DOS could be accessed if such portions of DOS were stored on a hard disk or floppy diskette. However, the invention is more than simply substituting a ROM for a disk because of several problems. A first thought that might occur to many persons is that by storing DOS in a ROM, DOS can then be executed directly from the ROM. However, while certain portions of DOS, such as commands in the COMMAND.COM program, can be executed directly from ROM, other portions, such as IBMBIO.COM and IBMDOS.COM are altered and cannot be used in a read only device which precludes any alteration.

Feigenbaum, 1:40-57

4. The method of claim 1, wherein the updating comprises: associating additional boot data with the boot data list; and compressing a portion of the additional boot data.	Feigenbaum, as evidenced by the example citations below, discloses "wherein the updating comprises: associating additional boot data with the boot data list; and compressing a portion of the additional boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein associating additional boot data with the boot data list; and compressing a portion of the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

See Claims 1.4.1, and 2 above.

> A data processing system, such as a personal computer, contains bootable DOS programs that are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation.

Feigenbaum, Abstract

> A still further object is to provide improved method and apparatus for rapidly initializing a personal computer when it is turned on, which method and apparatus uses an IFS to access DOS programs stored in a ROM in a packed format differing from the format used to store such programs in a FAT file system.

Feigenbaum, 3:47-52

> FIG. 3 also illustrates another facet of the invention which is that the data and programs of ROM DOS 34 are stored in ROM 16 in a file system using a packed format. In contrast, any data and files stored on disk 24, including those in RAM DOS 36, are stored in the conventional DOS FAT file system in which information is stored in clusters and sectors. In the FAT file system, if a particular program doesn't have the same number of bytes as are in a sector or cluster, space is wasted. On the average, about one half the number of bytes in a cluster or in a sector are wasted for each file. Since ROM units are more expensive than disk storage, any wasted space is inefficient. Thus, in ROM 16, the ROM DOS 34 files are stored beginning at a segment address, each segment being sixteen bytes. Each succeeding file begins immediately at the next segment location following the end of a preceding file so that on average only

> eight bytes per file would be wasted, such number being far less than the average waste in a FAT file system. By way of example, suppose file 50 begins at ROM segment X and ends at Y. Then, the succeeding file 52 begins at segment Y+n, where n is less than sixteen. The other files are similarly stored.

Feigenbaum, 6:60-7:15

> Briefly, in accordance with the subject invention, bootable DOS programs are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation.

Feigenbaum, 3:53-59

> FIG. 4 illustrates the relative position and use of IFS HANDLER 53. Within the prior art, an application program 110 can be thought of as sitting on top of FAT file system 112, which in turn sits on top of disk/diskette driver 114 which overlies the disk 24 containing the actual files and data. Application 110 uses interrupt INT 21H to access the file system through IBMDOS.COM, which acts through IBMBIO.COM to access the disk. In accordance with the invention, IFS HANDLER 53 is interposed at the file system level in the INT 21H process and decides in step 116 whether the desired file system access is to be made to ROM DISK D:. If not, the system proceeds as in the prior art to access the FAT file system. If the desired access is to be made to ROM DISK D:, it is done through the ROM file system 118 and RAM LOADER 51 by programs IBMDOS.COM 52 and IBMBIO.COM 50. The term "ROM DISK" is defined hereby to mean a ROM unit containing images of files storable on a hard disk or floppy diskette, which images are stored in a packed format in a ROM file system, that is different from the FAT file system used to store files on a disk or diskette.

Feigenbaum, 9:39-60

5 (Preamble) A method for booting a computer system, the method comprising:	Feigenbaum, as evidenced by the example citations below, discloses "a method for booting a computer system, the method comprising:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

See Claims 1-1.4.2 above.

> A data processing system, such as a personal computer, contains bootable DOS programs that are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation.

Feigenbaum, Abstract

> When such personal computers are powered up, a power on self test (POST) routine or program is first executed, such program being stored in ROM. Upon the successful completion of such test, portions of DOS are read into the system memory from a hard disk or a floppy diskette and the system is booted up and initialized. The test, booting and initialization process can take many seconds. The present invention is directed to an improvement by which such process is significantly shortened in time. It is also common to start a personal computer by turning on not only a system unit but also a display, and the invention has an objective of accomplishing the start up process so that the system is available for use within the time required to warm up the display. That is, as soon as the system is turned on, the first screen becomes immediately visible on a display and gives the user the appearance of an "instant" load and initialization.

Feigenbaum, 1:22-39

> It is also known that the speed at which a ROM can be accessed is several orders of magnitude faster than the speed at which a hard disk or a floppy diskette can be accessed. The invention takes advantage of this known speed difference by storing portions of DOS in a ROM where such portions can be accessed at a speed much faster than the speed at which DOS could be accessed if such portions of DOS were stored on a hard disk or floppy diskette. However, the invention is more than simply substituting a ROM for a disk because of several problems. A first thought that might occur to many persons is that by

> storing DOS in a ROM, DOS can then be executed directly from the ROM. However, while certain portions of DOS, such as commands in the COMMAND.COM program, can be executed directly from ROM, other portions, such as IBMBIO.COM and IBMDOS.COM are altered and cannot be used in a read only device which precludes any alteration.

Feigenbaum, 1:40-57

> System 10 is further provided with a DOS 32 that comprises two major portions, ROM DOS 34 and DISK DOS 36 respectively stored in ROM 16 and disk 24. ROM DOS 34 includes the programs that are booted into the RAM when the system is initially powered on or when the system is reset, which programs provide the minimum level of operating system support to make system 10 operational to the user. The remaining portions of DOS, i.e. those programs and files that together form a complete DOS, are included in DISK DOS 36. DISK DOS 36 further includes DOS programs similar to those in ROM DOS so that the system can be booted up and operated from disk 24 at the option of the user, as described in more detail hereafter. Also stored in ROM 16 are a basic input/output system (BIOS) 38 and a power on self test (POST) routine 40. When system 10 is first turned on, or when it is restarted, POST 40 first performs the test and upon successful completion, it boots or loads a portion of DOS and transfers control to it in the manner described in detail below. Except for ROM DOS 34, the system as thus far described is constructed and operates in accordance with known principles of the prior art.

Feigenbaum, 5:3-25

> With reference to FIG. 2, a memory map 39 is illustrated based on the full address space of sixteen megabytes (MB) using the twenty-four bit addressing capability of microprocessor 12. The lowest 640 kilobytes (KB) are assigned to RAM 14 and form a storage area, known as the DOS address space, for containing DOS programs in the lowest portion thereof and application programs in the upper or top portion thereof. Addresses immediately below the one megabyte (MB) region are assigned to that portion of ROM 20 containing POST 38 and BIOS 40. A video buffer occupies the space immediately above the DOS address space beginning at hex address A0000 H. ROM DOS 34 occupies a 256KB region at the top of the 16 MB address space beginning at hex address FC0000 H. Thus, the lowest one MB region is assigned to the same addresses and functions as such region is used in the prior art, while ROM DOS 34 is assigned to an address space which is not immediately addressable by DOS nor application programs running in the DOS address space. The remaining regions are unused memory addresses. Further, the low 1MB address range is accessible in the real mode of operation of microprocessor 12, while the address range above the real mode range is accessible in protected mode.

Feigenbaum, 5:26-49

> The steps are performed by microprocessor 12 operating under program control to control the operation of the various components within system 10.

Feigenbaum, 8:34-37

5.1 storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory;	Feigenbaum, as evidenced by the example citations below, discloses "storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1.1 above. > Referring now to the drawings, and first to FIG. 1, the invention is embodied in a personal computer system 10, and resides in the manner in which such system is programmed and operated. It is to be appreciated that such computers are complex and include many components and data processing devices, such as device controllers and adapters, which have been omitted from the drawings for simplicity of illustration. The description provided herein is limited to only those items which are useful in understanding the invention. System 10 includes a microprocessor 12, such as an Intel 80286 microprocessor, which is commercially available and functions in a known manner to execute programs stored in a RAM 14 and a ROM 16. Such ROM preferably comprises a plurality of ROM units which together form ROM 16 and provide sufficient storage capacity for all of the stored information described in more detail below. Feigenbaum, 4:10-27	

5.2 loading the stored compressed boot data from the first memory;	Feigenbaum, as evidenced by the example citations below, discloses "loading the stored compressed boot data from the first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the stored compressed boot data from the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1.1 above.	

5.3 accessing the loaded compressed boot data;	Feigenbaum, as evidenced by the example citations below, discloses "accessing the loaded compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1.2 above.	

5.4 decompressing the accessed compressed boot data;	Feigenbaum, as evidenced by the example citations below, discloses “decompressing the accessed compressed boot data”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1.3 above.	

5.5 utilizing the decompressed boot data to at least partially boot the computer system;	Feigenbaum, as evidenced by the example citations below, discloses "utilizing the decompressed boot data to at least partially boot the computer system"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed boot data to at least partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

See Claims 1.3 and 1.4.2 above.

> When such personal computers are powered up, a power on self test (POST) routine or program is first executed, such program being stored in ROM. Upon the successful completion of such test, portions of DOS are read into the system memory from a hard disk or a floppy diskette and the system is booted up and initialized. The test, booting and initialization process can take many seconds. The present invention is directed to an improvement by which such process is significantly shortened in time. It is also common to start a personal computer by turning on not only a system unit but also a display, and the invention has an objective of accomplishing the start up process so that the system is available for use within the time required to warm up the display. That is, as soon as the system is turned on, the first screen becomes immediately visible on a display and gives the user the appearance of an "instant" load and initialization.

Feigenbaum, 1:22-39

> One of the objects of the invention is to provide a ROM based DOS that rapidly tests, boots, and initializes a personal computer and appears to the user to "instantly" start up when the computer is turned on.

Feigenbaum, 3:30-33

> When switch 35 is initially turned on, display 30 warms up within a relatively short period and the ROM based booting and system initialization, described in detail below, occurs within the period required for the display to warm up so that at the end of such period, the display screen becomes visible to the user to give the appearance of an instant startup.

Feigenbaum, 4:63-5:2

In summary, the invention of a ROM based DOS provides several features and advantages. DOS is rapidly loaded from ROM and executed in RAM to instantly boot up and present the user with a screen from a graphical user interface. Once booted, ROM DOS executes normally and acts the same as standard disk DOS and has the same system compatibility as disk DOS. Further, no DOS installation is required to be done by a user, and the user has no diskette dependencies. The invention is further advantageous by virtue of using the IFS interface and allowing the ROM to appear as a "drive", since this minimizes the number of changes to the basic DOS programs, which reduced developement time and minimized the possibility of error being introduced by new code. Feigenbaum, 10:18-33

5.6 updating the boot data list;	Feigenbaum, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1.4.1 above.	

5.7 wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form.	Feigenbaum, as evidenced by the example citations below, discloses "wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

See Claims 1-1.3 above.

> A data processing system, such as a personal computer, contains bootable DOS programs that are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation.

Feigenbaum, Abstract

> When such personal computers are powered up, a power on self test (POST) routine or program is first executed, such program being stored in ROM. Upon the successful completion of such test, portions of DOS are read into the system memory from a hard disk or a floppy diskette and the system is booted up and initialized. The test, booting and initialization process can take many seconds. The present invention is directed to an improvement by which such process is significantly shortened in time. It is also common to start a personal computer by turning on not only a system unit but also a display, and the invention has an objective of accomplishing the start up process so that the system is available for use within the time required to warm up the display. That is, as soon as the system is turned on, the first screen becomes immediately visible on a display and gives the user the appearance of an "instant" load and initialization.

Feigenbaum, 1:22-39

> It is also known that the speed at which a ROM can be accessed is several orders

> of magnitude faster than the speed at which a hard disk or a floppy diskette can be accessed. The invention takes advantage of this known speed difference by storing portions of DOS in a ROM where such portions can be accessed at a speed much faster than the speed at which DOS could be accessed if such portions of DOS were stored on a hard disk or floppy diskette. However, the invention is more than simply substituting a ROM for a disk because of several problems. A first thought that might occur to many persons is that by storing DOS in a ROM, DOS can then be executed directly from the ROM. However, while certain portions of DOS, such as commands in the COMMAND.COM program, can be executed directly from ROM, other portions, such as IBMBIO.COM and IBMDOS.COM are altered and cannot be used in a read only device which precludes any alteration.

Feigenbaum, 1:40-57

> In summary, the invention of a ROM based DOS provides several features and advantages. DOS is rapidly loaded from ROM and executed in RAM to instantly boot up and present the user with a screen from a graphical user interface. Once booted, ROM DOS executes normally and acts the same as standard disk DOS and has the same system compatibility as disk DOS. Further, no DOS installation is required to be done by a user, and the user has no diskette dependencies. The invention is further advantageous by virtue of using the IFS interface and allowing the ROM to appear as a "drive", since this minimizes the number of changes to the basic DOS programs, which reduced developement time and minimized the possibility of error being introduced by new code.

Feigenbaum, 10:18-33

6 (Preamble) A system comprising: a processor;	Feigenbaum, as evidenced by the example citations below, discloses “a system comprising: a processor:”

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system comprising: a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.

Feigenbaum discloses this limitation:

> A data processing system, such as a personal computer, contains bootable DOS programs that are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation.

Feigenbaum, Abstract

> Referring now to the drawings, and first to FIG. 1, the invention is embodied in a personal computer system 10, and resides in the manner in which such system is programmed and operated. It is to be appreciated that such computers are complex and include many components and data processing devices, such as device controllers and adapters, which have been omitted from the drawings for simplicity of illustration. The description provided herein is limited to only those items which are useful in understanding the invention. System 10 includes a microprocessor 12, such as an Intel 80286 microprocessor, which is commercially available and functions in a known manner to execute programs stored in a RAM 14 and a ROM 16. Such ROM preferably comprises a plurality of ROM units which together form ROM 16 and provide sufficient storage capacity for all of the stored information described in more detail below.

Feigenbaum, 4:10-27

> With reference to FIG. 2, a memory map 39 is illustrated based on the full address space of sixteen megabytes (MB) using the twenty-four bit addressing capability of microprocessor 12. The lowest 640 kilobytes (KB) are assigned to RAM 14 and form a storage area, known as the DOS address space, for containing DOS programs in the lowest portion thereof and application programs in the upper or top portion thereof. Addresses immediately below the one megabyte (MB) region are assigned to that portion of ROM 20 containing POST 38 and BIOS 40. A video buffer occupies the space immediately above the DOS address space beginning at hex address A0000 H. ROM DOS 34 occupies a 256KB region at the top of the 16 MB address space beginning at hex address FC0000 H. Thus,

> the lowest one MB region is assigned to the same addresses and functions as such region is used in the prior art, while ROM DOS 34 is assigned to an address space which is not immediately addressable by DOS nor application programs running in the DOS address space. The remaining regions are unused memory addresses. Further, the low 1MB address range is accessible in the real mode of operation of microprocessor 12, while the address range above the real mode range is accessible in protected mode.

Feigenbaum, 5:26-49

> The steps are performed by microprocessor 12 operating under program control to control the operation of the various components within system 10.

Feigenbaum, 8:34-37

6.1 a memory; and	Feigenbaum, as evidenced by the example citations below, discloses "a memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a memory.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 1.1 and 1.2 above.	

6.2 a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor	Feigenbaum, as evidenced by the example citations below, discloses "a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

See Claims 1.1 and 1.2 above.

> A data processing system, such as a personal computer, contains bootable DOS programs that are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation.

Feigenbaum, Abstract

> Referring now to the drawings, and first to FIG. 1, the invention is embodied in a personal computer system 10, and resides in the manner in which such system is programmed and operated. It is to be appreciated that such computers are complex and include many components and data processing devices, such as device controllers and adapters, which have been omitted from the drawings for simplicity of illustration. The description provided herein is limited to only those items which are useful in understanding the invention. System 10 includes a microprocessor 12, such as an Intel 80286 microprocessor, which is commercially available and functions in a known manner to execute programs stored in a RAM 14 and a ROM 16. Such ROM preferably comprises a plurality of ROM units which together form ROM 16 and provide sufficient storage capacity for all of the stored information described in more detail below.

Feigenbaum, 4:10-27

> With reference to FIG. 2, a memory map 39 is illustrated based on the full address space of sixteen megabytes (MB) using the twenty-four bit addressing capability of microprocessor 12. The lowest 640 kilobytes (KB) are assigned to RAM 14 and form a storage area, known as the DOS address space, for containing DOS

programs in the lowest portion thereof and application programs in the upper or top portion thereof. Addresses immediately below the one megabyte (MB) region are assigned to that portion of ROM 20 containing POST 38 and BIOS 40. A video buffer occupies the space immediately above the DOS address space beginning at hex address A0000 H. ROM DOS 34 occupies a 256KB region at the top of the 16 MB address space beginning at hex address FC0000 H. Thus, the lowest one MB region is assigned to the same addresses and functions as such region is used in the prior art, while ROM DOS 34 is assigned to an address space which is not immediately addressable by DOS nor application programs running in the DOS address space. The remaining regions are unused memory addresses. Further, the low 1MB address range is accessible in the real mode of operation of microprocessor 12, while the address range above the real mode range is accessible in protected mode.

Feigenbaum, 5:26-49

The steps are performed by microprocessor 12 operating under program control to control the operation of the various components within system 10.

Feigenbaum, 8:34-37

Another object is to store a DOS kernel in a ROM in a simple, efficient file structure that can be rapidly accessed using less code than that required to access a DOS kernel stored on a disk in a FAT file system.

Feigenbaum, 3:34-37

A still further object is to provide improved method and apparatus for rapidly initializing a personal computer when it is turned on, which method and apparatus uses an IFS to access DOS programs stored in a ROM in a packed format differing from the format used to store such programs in a FAT file system.

Feigenbaum, 3:47-52

Programs 50 through 56 provide the minimum level of operating system support for operating data processing system 10. IBMBIO.COM 50 (except for RAM LOADER 51), IBMDOS.COM 52 (except for IFS handler 53), and COMMANDS 54 AND 56 (which together form the conventional COMMAND.COM DOS program) comprise the conventional DOS kernel. IBMBIO.COM provides low level device support. IBMDOS.COM provides application support. COMMAND.COM is the command processor. COMMAND 54 stays resident in RAM 14 once it is loaded while COMMAND 56 can be overlaid and later copied into RAM 14 as needed. ROMSHELL 57 provides a graphical user interface and is transient while SHELSTUB 59 is designed to be resident in the RAM for reloading the ROMSHELL program after quitting an application. The ROM DOS 34 programs when loaded into the RAM generally operate the same as the corresponding programs in DISK DOS 36, the only difference being that such programs are loaded from ROM 16 instead of disk 24. Relative to other portions of DOS and the system, ROM DOS 34 is transparent and appears to act the same as corresponding portions of DISK DOS 36. The ROM DOS files are stored in ROM 16 outside of the address space available for DOS programs, and RAM LOADER 51 is able to work outside such address space to copy such programs into the DOS address space in RAM 14. RAM LOADER 51 uses a standard method of moving blocks of data from the protected mode address space into the real mode DOS address space, such as by using the well known system service interrupt 15H of BIOS 40 which is invoked by setting the well known register AH (not shown) of microprocessor 12 to the MOVE BLOCK function 87H. Such function transfers a block of data from storage above the IMB protected mode address range by switching to the protected mode. RAM LOADER 51 is invoked using interrupt 2BH and accesses the information from items 42 and 46 to load the ROM header at initialization time, check for the existence of a module in ROM, load a module from ROM, and access flag 58 in CMOS RAM 20.

Feigenbaum, 6:3-6:44

FIG. 4 illustrates the relative position and use of IFS HANDLER 53. Within the prior art, an application program 110 can be thought of as sitting on top of FAT file system 112, which in turn sits on top of disk/diskette driver 114 which overlies the disk 24 containing the actual files and data. Application 110 uses interrupt INT 21H to access the file system through IBMDOS.COM, which acts through IBMBIO.COM to access the disk. In accordance with the invention, IFS HANDLER 53 is interposed at the file system level in the INT 21H process and decides in step 116 whether the desired file system access is to be made to ROM DISK D:. If not, the system proceeds as in the prior art to access the FAT file system. If the desired access is to be made to ROM DISK D:, it is done through the ROM file system 118 and RAM LOADER 51 by programs IBMDOS.COM 52 and IBMBIO.COM 50. The term "ROM DISK" is defined hereby to mean a ROM unit containing images of files storable on a hard disk or floppy diskette, which images are stored in a packed format in a ROM file system, that is different from the FAT file system used to store files on a disk or diskette.

Feigenbaum, 9:39-60

The IFS interface "connects" the ROM as an installable file system and assigns it a drive letter to be referenced. Such interface performs calls at a file system level as opposed to other systems which operate at the disk level. Examples of interfaces operating at the system level are OPEN, READ, CLOSE, etc. Examples of interfaces which operate at the disk level are READ SECTOR, and WRITE SECTOR. Operating system performance is improved since file system

> calls are identified earlier in the code path than disk calls. Further, in view of the relative simplicity of the ROM file structure in comparison to the complex file structure of the FAT file system, it is obvious that the amount of code necessary to access the ROM file system is substantially less than the amount of code necessary to access the FAT file system. Thus performance is faster because less code has to be executed and because the FAT file system code is bypassed due to the system level interrupt being located in the code path prior to the FAT file system code in IBMDOS.COM.

Feigenbaum, 9:61-10:12

6.3 wherein the processor is configured:	Feigenbaum, as evidenced by the example citations below, discloses “wherein the processor is configured:”

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the processor is configured), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.

Feigenbaum discloses this limitation:

> A data processing system, such as a personal computer, contains bootable DOS programs that are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation.

Feigenbaum, Abstract

> Referring now to the drawings, and first to FIG. 1, the invention is embodied in a personal computer system 10, and resides in the manner in which such system is programmed and operated. It is to be appreciated that such computers are complex and include many components and data processing devices, such as device controllers and adapters, which have been omitted from the drawings for simplicity of illustration. The description provided herein is limited to only those items which are useful in understanding the invention. System 10 includes a microprocessor 12, such as an Intel 80286 microprocessor, which is commercially available and functions in a known manner to execute programs stored in a RAM 14 and a ROM 16. Such ROM preferably comprises a plurality of ROM units which together form ROM 16 and provide sufficient storage capacity for all of the stored information described in more detail below.

Feigenbaum, 4:10-27

> With reference to FIG. 2, a memory map 39 is illustrated based on the full address space of sixteen megabytes (MB) using the twenty-four bit addressing capability of microprocessor 12. The lowest 640 kilobytes (KB) are assigned to RAM 14 and form a storage area, known as the DOS address space, for containing DOS programs in the lowest portion thereof and application programs in the upper or top portion thereof. Addresses immediately below the one megabyte (MB) region are assigned to that portion of ROM 20 containing POST 38 and BIOS 40. A video buffer occupies the space immediately above the DOS address space beginning at hex address A0000 H. ROM DOS 34 occupies a 256KB region at

> the top of the 16 MB address space beginning at hex address FC0000 H. Thus, the lowest one MB region is assigned to the same addresses and functions as such region is used in the prior art, while ROM DOS 34 is assigned to an address space which is not immediately addressable by DOS nor application programs running in the DOS address space. The remaining regions are unused memory addresses. Further, the low 1MB address range is accessible in the real mode of operation of microprocessor 12, while the address range above the real mode range is accessible in protected mode.

Feigenbaum, 5:26-49

> The steps are performed by microprocessor 12 operating under program control to control the operation of the various components within system 10.

Feigenbaum, 8:34-37

6.4 to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory,	Feigenbaum, as evidenced by the example citations below, discloses “to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1.1 above.	

6.5 to access the loaded portion of the boot data in the compressed form,	Feigenbaum, as evidenced by the example citations below, discloses "to access the loaded portion of the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to access the loaded portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1.2 above.	

6.6 to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data, and	Feigenbaum, as evidenced by the example citations below, discloses “to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1.3 above.	

6.7 to update the boot data list.	Feigenbaum, as evidenced by the example citations below, discloses "to update the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to update the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1.4.1 above.	

Appendix C4
Invalidity of U.S. Patent 8,880,862 based on Feigenbaum

8 (Preamble) A method of loading an operating system for booting a computer system, comprising:	Feigenbaum, as evidenced by the example citations below, discloses "A method of loading an operating system for booting a computer system, comprising:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method of loading an operating system for booting a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1 (Preamble) above.	

8.1 storing a portion of the operating system in a compressed form in a first memory;	Feigenbaum, as evidenced by the example citations below, discloses “storing a portion of the operating system in a compressed form in a first memory”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing a portion of the operating system in a compressed form in a first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 6.2 above.	

8.2 loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list;	Feigenbaum, as evidenced by the example citations below, discloses “loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1.1 above.	

8.3 accessing the loaded portion of the operating system from the second memory in the compressed form;	Feigenbaum, as evidenced by the example citations below, discloses “accessing the loaded portion of the operating system from the second memory in the compressed form”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded portion of the operating system from the second memory in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1.2 above.	

8.4 decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system;	Feigenbaum, as evidenced by the example citations below, discloses "decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 1.3 and 1.4.2 above.	

8.5 utilizing the decompressed portion of the operating system to at least partially boot the computer system; and	Feigenbaum, as evidenced by the example citations below, discloses “utilizing the decompressed portion of the operating system to at least partially boot the computer system”

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed portion of the operating system to at least partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.

Feigenbaum discloses this limitation:

See Claims 1.3 and 1.4.2 above.

> When such personal computers are powered up, a power on self test (POST) routine or program is first executed, such program being stored in ROM. Upon the successful completion of such test, portions of DOS are read into the system memory from a hard disk or a floppy diskette and the system is booted up and initialized. The test, booting and initialization process can take many seconds. The present invention is directed to an improvement by which such process is significantly shortened in time. It is also common to start a personal computer by turning on not only a system unit but also a display, and the invention has an objective of accomplishing the start up process so that the system is available for use within the time required to warm up the display. That is, as soon as the system is turned on, the first screen becomes immediately visible on a display and gives the user the appearance of an "instant" load and initialization.

Feigenbaum, 1:22-39

> One of the objects of the invention is to provide a ROM based DOS that rapidly tests, boots, and initializes a personal computer and appears to the user to "instantly" start up when the computer is turned on.

Feigenbaum, 3:30-33

> When switch 35 is initially turned on, display 30 warms up within a relatively short period and the ROM based booting and system initialization, described in detail below, occurs within the period required for the display to warm up so that at the end of such period, the display screen becomes visible to the user to give the appearance of an instant startup.

Feigenbaum, 4:63-5:2

> In summary, the invention of a ROM based DOS provides several features and advantages. DOS is rapidly loaded from ROM and executed in RAM to instantly boot up and present the user with a screen from a graphical user interface. Once booted, ROM DOS executes normally and acts the same as standard disk DOS and has the same system compatibility as disk DOS. Further, no DOS installation is required to be done by a user, and the user has no diskette dependencies. The invention is further advantageous by virtue of using the IFS interface and allowing the ROM to appear as a "drive", since this minimizes the number of changes to the basic DOS programs, which reduced developement time and minimized the possibility of error being introduced by new code.

Feigenbaum, 10:18-33

8.6 updating the boot data list,	Feigenbaum, as evidenced by the example citations below, discloses “updating the boot data list”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1.4.1 above.	

8.7 wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form.	Feigenbaum, as evidenced by the example citations below, discloses "wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

See Claims 1-1.3 and 5.7 above.

> A data processing system, such as a personal computer, contains bootable DOS programs that are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation.

Feigenbaum, Abstract

> When such personal computers are powered up, a power on self test (POST) routine or program is first executed, such program being stored in ROM. Upon the successful completion of such test, portions of DOS are read into the system memory from a hard disk or a floppy diskette and the system is booted up and initialized. The test, booting and initialization process can take many seconds. The present invention is directed to an improvement by which such process is significantly shortened in time. It is also common to start a personal computer by turning on not only a system unit but also a display, and the invention has an objective of accomplishing the start up process so that the system is available for use within the time required to warm up the display. That is, as soon as the system is turned on, the first screen becomes immediately visible on a display and gives

> the user the appearance of an "instant" load and initialization.

Feigenbaum, 1:22-39

> It is also known that the speed at which a ROM can be accessed is several orders of magnitude faster than the speed at which a hard disk or a floppy diskette can be accessed. The invention takes advantage of this known speed difference by storing portions of DOS in a ROM where such portions can be accessed at a speed much faster than the speed at which DOS could be accessed if such portions of DOS were stored on a hard disk or floppy diskette. However, the invention is more than simply substituting a ROM for a disk because of several problems. A first thought that might occur to many persons is that by storing DOS in a ROM, DOS can then be executed directly from the ROM. However, while certain portions of DOS, such as commands in the COMMAND.COM program, can be executed directly from ROM, other portions, such as IBMBIO.COM and IBMDOS.COM are altered and cannot be used in a read only device which precludes any alteration.

Feigenbaum, 1:40-57

> In summary, the invention of a ROM based DOS provides several features and advantages. DOS is rapidly loaded from ROM and executed in RAM to instantly boot up and present the user with a screen from a graphical user interface. Once booted, ROM DOS executes normally and acts the same as standard disk DOS and has the same system compatibility as disk DOS. Further, no DOS installation is required to be done by a user, and the user has no diskette dependencies. The invention is further advantageous by virtue of using the IFS interface and allowing the ROM to appear as a "drive", since this minimizes the number of changes to the basic DOS programs, which reduced developement time and minimized the possibility of error being introduced by new code.

Feigenbaum, 10:18-33

Appendix C4
Invalidity of U.S. Patent 8,880,862 based on Feigenbaum

9.1 The method of claim 8, further comprising: compressing an additional portion of the operating system that is not associated with the boot data list; and	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 8, further comprising: compressing an additional portion of the operating system that is not associated with the boot data list"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, compressing an additional portion of the operating system that is not associated with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

See Claim 1.1 above.

> It is also known that the speed at which a ROM can be accessed is several orders of magnitude faster than the speed at which a hard disk or a floppy diskette can be accessed. The invention takes advantage of this known speed difference by storing portions of DOS in a ROM where such portions can be accessed at a speed much faster than the speed at which DOS could be accessed if such portions of DOS were stored on a hard disk or floppy diskette. However, the invention is more than simply substituting a ROM for a disk because of several problems. A first thought that might occur to many persons is that by storing DOS in a ROM, DOS can then be executed directly from the ROM. However, while certain portions of DOS, such as commands in the COMMAND.COM program, can be executed directly from ROM, other portions, such as IBMBIO.COM and IBMDOS.COM are altered and cannot be used in a read only device which precludes any alteration.

Feigenbaum, 1:40-57

9.2 storing the additional portion of the operating system in the first memory, and	Feigenbaum, as evidenced by the example citations below, discloses "storing the additional portion of the operating system in the first memory"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing the additional portion of the operating system in the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

See Claim 8.1 above.

> A further object is to provide a ROM based DOS which uses an IFS that is accessible at the system level, as opposed to the device level, to rapidly load DOS programs from ROM into RAM for execution from such RAM.

Feigenbaum, 3:38-42

> Still another object is to provide a ROM based DOS in which at least certain DOS programs that are stored on a disk or diskette in a FAT file system, are stored in the ROM using an alternate file system.

Feigenbaum, 3:43-46

> A still further object is to provide improved method and apparatus for rapidly initializing a personal computer when it is turned on, which method and apparatus uses an IFS to access DOS programs stored in a ROM in a packed format differing from the format used to store such programs in a FAT file system.

Feigenbaum, 3:47-52

> System 10 is further provided with a DOS 32 that comprises two major portions, ROM DOS 34 and DISK DOS 36 respectively stored in ROM 16 and disk 24. ROM DOS 34 includes the programs that are booted into the RAM when the system is initially powered on or when the system is reset, which programs provide the minimum level of operating system support to make system 10 operational to the user. The remaining portions of DOS, i.e. those programs and files that together form a complete DOS, are included in DISK DOS 36. DISK DOS 36 further includes DOS programs similar to those in ROM DOS so that the system can be booted up and operated from disk 24 at the option of the user, as described in more detail hereafter. Also stored in

> ROM 16 are a basic input/output system (BIOS) 38 and a power on self test (POST) routine 40. When system 10 is first turned on, or when it is restarted, POST 40 first performs the test and upon successful completion, it boots or loads a portion of DOS and transfers control to it in the manner described in detail below. Except for ROM DOS 34, the system as thus far described is constructed and operates in accordance with known principles of the prior art.

Feigenbaum, 5:3-25

Appendix C4
Invalidity of U.S. Patent 8,880,862 based on Feigenbaum

9.3 wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system.	Feigenbaum, as evidenced by the example citations below, discloses "wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

See Claim 8.5 above.

> System 10 is further provided with a DOS 32 that comprises two major portions, ROM DOS 34 and DISK DOS 36 respectively stored in ROM 16 and disk 24. ROM DOS 34 includes the programs that are booted into the RAM when the system is initially powered on or when the system is reset, which programs provide the minimum level of operating system support to make system 10 operational to the user. The remaining portions of DOS, i.e. those programs and files that together form a complete DOS, are included in DISK DOS 36. DISK DOS 36 further includes DOS programs similar to those in ROM DOS so that the system can be booted up and operated from disk 24 at the option of the user, as described in more detail hereafter. Also stored in ROM 16 are a basic input/output system (BIOS) 38 and a power on self test (POST) routine 40. When system 10 is first turned on, or when it is restarted, POST 40 first performs the test and upon successful completion, it boots or loads a portion of DOS and transfers control to it in the manner described in detail below. Except for ROM DOS 34, the system as thus far described is constructed and operates in accordance with known principles of the prior art.

Feigenbaum, 5:3-25

Appendix C4
Invalidity of U.S. Patent 8,880,862 based on Feigenbaum

10. The method of claim 9, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 9, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

See Claim 9.1 above.

> It is also known that the speed at which a ROM can be accessed is several orders of magnitude faster than the speed at which a hard disk or a floppy diskette can be accessed. The invention takes advantage of this known speed difference by storing portions of DOS in a ROM where such portions can be accessed at a speed much faster than the speed at which DOS could be accessed if such portions of DOS were stored on a hard disk or floppy diskette. However, the invention is more than simply substituting a ROM for a disk because of several problems. A first thought that might occur to many persons is that by storing DOS in a ROM, DOS can then be executed directly from the ROM. However, while certain portions of DOS, such as commands in the COMMAND.COM program, can be executed directly from ROM, other portions, such as IBMBIO.COM and IBMDOS.COM are altered and cannot be used in a read only device which precludes any alteration.

Feigenbaum, 1:40-57

Appendix C4
Invalidity of U.S. Patent 8,880,862 based on Feigenbaum

11 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	Feigenbaum, as evidenced by the example citations below, discloses “a method for providing accelerated loading of an operating system in a computer system, comprising:”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1 (Preamble) above.	

11.1 loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system;	Feigenbaum, as evidenced by the example citations below, discloses "loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1.1 above. > A data processing system, such as a personal computer, contains bootable DOS programs that are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation. Feigenbaum, Abstract > Referring now to the drawings, and first to FIG. 1, the invention is embodied in a personal computer system 10, and resides in the manner in which such system is programmed and operated. It is to be appreciated that such computers are complex and include many components and data processing devices, such as device controllers and adapters, which have been omitted from the drawings for simplicity of illustration. The description provided herein is limited to only those items which are useful in understanding the invention. System 10 includes a microprocessor 12, such as an Intel 80286 microprocessor, which is commercially available and functions in a known manner to execute programs stored in a RAM 14 and a ROM 16. Such ROM preferably comprises a plurality of ROM units which together form ROM 16 and provide sufficient storage capacity for all of the stored information described in more detail below. Feigenbaum, 4:10-27 > With reference to FIG. 2, a memory map 39 is illustrated based on the full address space of sixteen megabytes (MB) using the twenty-four bit addressing capability of microprocessor 12. The lowest 640 kilobytes (KB) are assigned to RAM 14	

> and form a storage area, known as the DOS address space, for containing DOS programs in the lowest portion thereof and application programs in the upper or top portion thereof. Addresses immediately below the one megabyte (MB) region are assigned to that portion of ROM 20 containing POST 38 and BIOS 40. A video buffer occupies the space immediately above the DOS address space beginning at hex address A0000 H. ROM DOS 34 occupies a 256KB region at the top of the 16 MB address space beginning at hex address FC0000 H. Thus, the lowest one MB region is assigned to the same addresses and functions as such region is used in the prior art, while ROM DOS 34 is assigned to an address space which is not immediately addressable by DOS nor application programs running in the DOS address space. The remaining regions are unused memory addresses. Further, the low 1MB address range is accessible in the real mode of operation of microprocessor 12, while the address range above the real mode range is accessible in protected mode.

Feigenbaum, 5:26-49

> The steps are performed by microprocessor 12 operating under program control to control the operation of the various components within system 10.

Feigenbaum, 8:34-37

11.2 accessing the loaded boot data in compressed form from the memory;	Feigenbaum, as evidenced by the example citations below, discloses “accessing the loaded boot data in compressed form from the memory”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded boot data in compressed form from the memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1.2 above.	

11.3 decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form;	Feigenbaum, as evidenced by the example citations below, discloses "decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1.3 above.	

11.4 utilizing the decompressed boot data to load at least a portion of the operating system for the computer system; and	Feigenbaum, as evidenced by the example citations below, discloses “utilizing the decompressed boot data to load at least a portion of the operating system for the computer system”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed boot data to load at least a portion of the operating system for the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Feigenbaum discloses this limitation: When such personal computers are powered up, a power on self test (POST) routine or program is first executed, such program being stored in ROM. Upon the successful completion of such test, portions of DOS are read into the system memory from a hard disk or a floppy diskette and the system is booted up and initialized. The test, booting and initialization process can take many seconds. The present invention is directed to an improvement by which such process is significantly shortened in time. It is also common to start a personal computer by turning on not only a system unit but also a display, and the invention has an objective of accomplishing the start up process so that the system is available for use within the time required to warm up the display. That is, as soon as the system is turned on, the first screen becomes immediately visible on a display and gives the user the appearance of an "instant" load and initialization. Feigenbaum, 1:22-39 One of the objects of the invention is to provide a ROM based DOS that rapidly tests, boots, and initializes a personal computer and appears to the user to "instantly" start up when the computer is turned on. Feigenbaum, 3:30-33 When switch 35 is initially turned on, display 30 warms up within a relatively short period and the ROM based booting and system initialization, described in detail below, occurs within the period required for the display to warm up so that at the end of such period, the display screen becomes visible to the user to give the appearance of an instant startup. Feigenbaum, 4:63-5:2	

> In summary, the invention of a ROM based DOS provides several features and advantages. DOS is rapidly loaded from ROM and executed in RAM to instantly boot up and present the user with a screen from a graphical user interface. Once booted, ROM DOS executes normally and acts the same as standard disk DOS and has the same system compatibility as disk DOS. Further, no DOS installation is required to be done by a user, and the user has no diskette dependencies. The invention is further advantageous by virtue of using the IFS interface and allowing the ROM to appear as a "drive", since this minimizes the number of changes to the basic DOS programs, which reduced developement time and minimized the possibility of error being introduced by new code.

Feigenbaum, 10:18-33

11.5 updating the boot data list.	Feigenbaum, as evidenced by the example citations below, discloses “updating the boot data list”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1.4.1 above.	

Appendix C4
Invalidity of U.S. Patent 8,880,862 based on Feigenbaum

13 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	Feigenbaum, as evidenced by the example citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, comprising"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1 (Preamble) above.	

13.1 loading boot data in a compressed form that is associated with a boot data list from a boot device;	Feigenbaum, as evidenced by the example citations below, discloses “loading boot data in a compressed form that is associated with a boot data list from a boot device”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading boot data in a compressed form that is associated with a boot data list from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1.1 above.	

13.2 accessing the loaded boot data in the compressed form;	Feigenbaum, as evidenced by the example citations below, discloses “accessing the loaded boot data in the compressed form”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1.2 above.	

13.3 decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form;	Feigenbaum, as evidenced by the example citations below, discloses "decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1.3 above.	

13.4 updating the boot data list.	Feigenbaum, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1.4.1 above.	

Appendix C4
Invalidity of U.S. Patent 8,880,862 based on Feigenbaum

14 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	Feigenbaum, as evidenced by the example citations below, discloses “a method for providing accelerated loading of an operating system in a computer system, comprising”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1 (Preamble) above.	

14.1 accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list;	Feigenbaum, as evidenced by the example citations below, discloses "accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

See Claims 1.1 and 1.2 above.

> A data processing system, such as a personal computer, contains bootable DOS programs that are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation.

Feigenbaum, Abstract

> When such personal computers are powered up, a power on self test (POST) routine or program is first executed, such program being stored in ROM. Upon the successful completion of such test, portions of DOS are read into the system memory from a hard disk or a floppy diskette and the system is booted up and initialized. The test, booting and initialization process can take many seconds. The present invention is directed to an improvement by which such process is significantly shortened in time. It is also common to start a personal computer by turning on not only a system unit but also a display, and the invention has an objective of accomplishing the start up process so that the system is available for use within the time required to warm up the display. That is, as soon as the system is turned on, the first screen becomes immediately visible on a display and gives the user the appearance of an "instant" load and initialization.

Feigenbaum, 1:22-39

> It is also known that the speed at which a ROM can be accessed is several orders of magnitude faster than the speed at which a hard disk or a floppy diskette can be accessed. The invention takes advantage of this known speed difference by storing portions of DOS in a ROM where such portions can be

> accessed at a speed much faster than the speed at which DOS could be accessed if such portions of DOS were stored on a hard disk or floppy diskette. However, the invention is more than simply substituting a ROM for a disk because of several problems. A first thought that might occur to many persons is that by storing DOS in a ROM, DOS can then be executed directly from the ROM. However, while certain portions of DOS, such as commands in the COMMAND.COM program, can be executed directly from ROM, other portions, such as IBMBIO.COM and IBMDOS.COM are altered and cannot be used in a read only device which precludes any alteration.

Feigenbaum, 1:40-57

> System 10 is further provided with a DOS 32 that comprises two major portions, ROM DOS 34 and DISK DOS 36 respectively stored in ROM 16 and disk 24. ROM DOS 34 includes the programs that are booted into the RAM when the system is initially powered on or when the system is reset, which programs provide the minimum level of operating system support to make system 10 operational to the user. The remaining portions of DOS, i.e. those programs and files that together form a complete DOS, are included in DISK DOS 36. DISK DOS 36 further includes DOS programs similar to those in ROM DOS so that the system can be booted up and operated from disk 24 at the option of the user, as described in more detail hereafter. Also stored in ROM 16 are a basic input/output system (BIOS) 38 and a power on self test (POST) routine 40. When system 10 is first turned on, or when it is restarted, POST 40 first performs the test and upon successful completion, it boots or loads a portion of DOS and transfers control to it in the manner described in detail below. Except for ROM DOS 34, the system as thus far described is constructed and operates in accordance with known principles of the prior art.

Feigenbaum, 5:3-25

14.2 loading the boot data into a memory; and	Feigenbaum, as evidenced by the example citations below, discloses "loading the boot data into a memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the boot data into a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1.1 above.	

14.3 servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form; and	Feigenbaum, as evidenced by the example citations below, discloses "servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

See Claims 1.2 and 1.3 above.

> A data processing system, such as a personal computer, contains bootable DOS programs that are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation.

Feigenbaum, Abstract

> When such personal computers are powered up, a power on self test (POST) routine or program is first executed, such program being stored in ROM. Upon the successful completion of such test, portions of DOS are read into the system memory from a hard disk or a floppy diskette and the system is booted up and initialized. The test, booting and initialization process can take many seconds. The present invention is directed to an improvement by which such process is significantly shortened in time. It is also common to start a personal computer by turning on not only a system unit but also a display, and the invention has an objective of accomplishing the start up process so that the system is available for use within the time required to warm up the display. That is, as soon as the system is turned on, the first screen becomes immediately visible on a display and gives the user the appearance of an "instant" load and initialization.

Feigenbaum, 1:22-39

> System 10 is further provided with a DOS 32 that comprises two major portions, ROM DOS 34 and DISK DOS 36 respectively stored in ROM 16 and disk 24. ROM DOS 34 includes the programs that are booted into the RAM when the system is initially powered on or when the system is reset, which programs provide the minimum level of operating system support to make system 10 operational to the user. The remaining portions of DOS, i.e. those programs and files that together form a complete DOS, are included in DISK DOS 36. DISK DOS 36 further includes DOS programs similar to those in ROM DOS so that the system can be booted up and operated from disk 24 at the option of the user, as described in more detail hereafter. Also stored in ROM 16 are a basic input/output system (BIOS) 38 and a power on self test (POST) routine 40. When system 10 is first turned on, or when it is restarted, POST 40 first performs the test and upon successful completion, it boots or loads a portion of DOS and transfers control to it in the manner described in detail below. Except for ROM DOS 34, the system as thus far described is constructed and operates in accordance with known principles of the prior art.

Feigenbaum, 5:3-25

> When system 10 is first setup, it is desireable to customize the system in a manner similar to that in which IBM PS/2 systems are configured. During the course of such customization the user has to make decisions and define what features are to be used. The primary decision pertinent to the invention is whether to boot the system from ROM, in accordance with the invention, or to boot the system from disk 24 in the manner of the prior art. The selection of the option is then coded into a flag 58 which is stored during customization in CMOS RAM 20. Other options for the user are whether the system will be started by using the ROM based CONFIG.SYS 62 and/or AUTOEXEC.BAT 61, or the user defined CONFIG.SYS 102 or AUTOEXEC.BAT 102 files stored on disk 24. Flag 58 will also be set to reflect such other options. Default values are provided for such options so that in the absence of selecting other options, the system will automatically boot from ROM 16 and process the AUTOEXEC.BAT and CONFIG.SYS files therein.

Feigenbaum, 7:15-35

14.4 updating the boot data list.	Feigenbaum, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 1.4.1 above.	

15. The method of claim 14, wherein the boot data comprises: a program code associated with the operating system.	Feigenbaum, as evidenced by the example citations below, discloses “the method of claim 14, wherein the boot data comprises: a program code associated with the operating system.”

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the boot data comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.

Feigenbaum discloses this limitation:

See Claims 1 (Preamble) and 1.1 above.

> ROMBOOT 65 first loads IBMBIO.COM 50, including RAM LOADER 51, from ROM 16 into RAM 14 by step 76. IBMBIO.COM includes two code segments, IBMINIT and SYSINIT. Step 78 then transfers control to IBMINIT 66. Step 80 initializes the system device drivers and hardware. Step 81 provides or sets up a hook into interrupt handler INT 2BH for the RAM LOADER. Step 82 relocates the initialization routine of IBMBIO.COM 50 to predetermined locations at the top of the DOS address space in RAM 16. Next, step 84 transfers control to SYSINIT 67 which first, by step 86, loads IBMDOS.COM 52 from ROM 16 into RAM 14. Step 88 then executes the initialization routine of IBMDOS.COM. Afterwards, step 90 attaches the IFS Handler as the next available drive which is drive D: because system 10 includes a fixed disk drive. Next, step 91 selects the ROM drive created by the IFS handler, as the defualt drive.

Feigenbaum, 8:53-9:2

> In summary, the invention of a ROM based DOS provides several features and advantages. DOS is rapidly loaded from ROM and executed in RAM to instantly boot up and present the user with a screen from a graphical user interface. Once booted, ROM DOS executes normally and acts the same as standard disk DOS and has the same system compatibility as disk DOS. Further, no DOS installation is required to be done by a user, and the user has no diskette dependencies. The invention is further advantageous by virtue of using the IFS interface and allowing the ROM to appear as a "drive", since this minimizes the number of changes to the basic DOS programs, which reduced developement time and minimized the possibility of error being introduced by new code.

Feigenbaum, 10:18-33

16. The method of claim 14, wherein the operating system comprises: a plurality of files.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 14, wherein the operating system comprises: a plurality of files."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

> A concept that has been only relatively recently introduced by IBM into the field of personal computers is that of an installable file system. An IFS was designed to allow alternate file systems to be used with DOS and OS/2. Previously, only the well known file allocation table (FAT) file system was supported. An IFS was a feature of DOS version 4.00 which was used for a network file system to interconnect a personal computer and a network server. An IFS was also included in OS/2 version 1.2 and used for the high performance file system (HPFS) designed for high performance, large capacity disk drives.

Feigenbaum, 2:4-15

> Another object is to store a DOS kernel in a ROM in a simple, efficient file structure that can be rapidly accessed using less code than that required to access a DOS kernel stored on a disk in a FAT file system.

Feigenbaum, 3:34-37

> A further object is to provide a ROM based DOS which uses an IFS that is accessible at the system level, as opposed to the device level, to rapidly load DOS programs from ROM into RAM for execution from such RAM.

Feigenbaum, 3:38-42

> Still another object is to provide a ROM based DOS in which at least certain DOS programs that are stored on a disk or diskette in a FAT file system, are stored in the ROM using an alternate file system.

Feigenbaum, 3:43-46

> Briefly, in accordance with the subject invention, bootable DOS programs are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation.

Feigenbaum, 3:53-59

> Referring to FIG. 3, ROM DOS 34 includes various files for loading the DOS kernel and starting system 10 with a shell program. The first 512 bytes comprise ROM identification information 42, a jump instruction 44, module information and data 46, and a ROM BOOT program 48. ROM DOS 34 further includes an IBMBIO.COM program 50 including a RAM LOADER 51, an IBMDOS.COM program 52 including an IFS handler 53, a resident COMMAND program 54, a transient COMMAND program 57, a SHELSTUB program 59, a ROMSHELL program 57, a mouse driver MOUSE.COM 60, an AUTOEXEC.BAT file 61, a CONFIG.SYS file 62, and a CHECKC.COM program 63. Items 42 through 48 are stored in the first 512 bytes of ROM DOS 34 and are comparable to the conventional bootstrap record stored in the first sector of disk 24. The final steps performed by POST 40 are to copy the first 512 bytes of ROM DOS 34 into RAM 14 and then to transfer control, by a jump instruction, to jump instruction 44 which in turn jumps to the start of ROM BOOT program 48.

Feigenbaum, 5:50-6:2

> FIG. 3 also illustrates another facet of the invention which is that the data and programs of ROM DOS 34 are stored in ROM 16 in a file system using a packed format. In contrast, any data and files stored on disk 24, including those in RAM DOS 36, are stored in the conventional DOS FAT file system in which information is stored in clusters and sectors. In the FAT file system, if a particular program doesn't have the same number of bytes as are in a sector or cluster, space is wasted. On the average, about one half the number of bytes in a cluster or in a sector are wasted for each file. Since ROM units are more expensive than disk storage, any wasted space is inefficient. Thus, in ROM 16, the ROM DOS 34 files are stored beginning at a segment address, each segment being sixteen bytes. Each succeeding file begins immediately at the next segment location following the end of a preceding file so that on average only eight bytes per file would be wasted, such number being far less than the average waste in a FAT file system. By way of example, suppose file 50 begins at ROM segment X and ends at Y. Then, the succeeding file 52 begins at segment Y+n, where n is less than sixteen. The other files are similarly stored.

Feigenbaum, 6:60-7:15

> The IFS interface "connects" the ROM as an installable file system and assigns it a drive letter to be referenced. Such interface performs calls at a file system level as opposed to other systems which operate at the disk level. Examples of interfaces operating at the system level are OPEN, READ, CLOSE, etc. Examples of interfaces which operate at the disk level are READ SECTOR, and WRITE SECTOR. Operating system performance is improved since file system calls are identified earlier in the code path than disk calls. Further, in view of the relative simplicity of the ROM file structure in comparison to the complex file structure of the FAT file system, it is obvious that the amount of code necessary to access the ROM file system is substantially less than the amount of code necessary to access the FAT file system. Thus performance is faster because less code has to be executed and because the FAT file system code is bypassed due to the system level interrupt being located in the code path prior to the FAT file system code in IBMDOS.COM.

Feigenbaum, 9:61-10:12

17. The method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 15 above. > In summary, the invention of a ROM based DOS provides several features and advantages. DOS is rapidly loaded from ROM and executed in RAM to instantly boot up and present the user with a screen from a graphical user interface. Once booted, ROM DOS executes normally and acts the same as standard disk DOS and has the same system compatibility as disk DOS. Further, no DOS installation is required to be done by a user, and the user has no diskette dependencies. The invention is further advantageous by virtue of using the IFS interface and allowing the ROM to appear as a "drive", since this minimizes the number of changes to the basic DOS programs, which reduced developement time and minimized the possibility of error being introduced by new code. Feigenbaum, 10:18-33	

Appendix C4
Invalidity of U.S. Patent 8,880,862 based on Feigenbaum

19.1 The method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

> A data processing system, such as a personal computer, contains bootable DOS programs that are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation.

Feigenbaum, Abstract

> Another solution that might occur would be to create the ROM within the DOS address space, or within the first one megabyte of memory mapped space, and then load or copy DOS into a RAM (random access memory) within such address space to occupy and execute from the same space in RAM and operate in the same manner as DOS currently does. The disadvantage of such a solution is that two copies of DOS would then exist in such limited address space, one copy being the unalterable ROM DOS and the other copy being the alterable one that is stored in and executed from the RAM. Having two copies of essentially the same program in such a limited address space would not be efficient use of memory nor acceptable by many users.

Feigenbaum, 1:58-2:3

> Briefly, in accordance with the subject invention, bootable DOS programs are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation.

Feigenbaum, 1:53-60

> Referring to FIG. 3, ROM DOS 34 includes various files for loading the DOS kernel and starting system 10 with a shell program. The first 512 bytes comprise ROM identification information 42, a jump instruction 44, module information and data 46, and a ROM BOOT program 48. ROM DOS 34 further includes an IBMBIO.COM program 50 including a RAM LOADER 51, an IBMDOS.COM program 52 including an IFS handler 53, a resident COMMAND program 54, a transient COMMAND program 57, a SHELSTUB program 59, a ROMSHELL program 57, a mouse driver MOUSE.COM 60, an AUTOEXEC.BAT file 61, a CONFIG.SYS file 62, and a CHECKC.COM program 63. Items 42 through 48 are stored in the first 512 bytes of ROM DOS 34 and are comparable to the conventional bootstrap record stored in the first sector of disk 24. The final steps performed by POST 40 are to copy the first 512 bytes of ROM DOS 34 into RAM 14 and then to transfer control, by a jump instruction, to jump instruction 44 which in turn jumps to the start of ROM BOOT program 48.

Feigenbaum, 5:50-6:2

> Programs 50 through 56 provide the minimum level of operating system support for operating data processing system 10. IBMBIO.COM 50 (except for RAM LOADER 51), IBMDOS.COM 52 (except for IFS handler 53), and COMMANDS 54 AND 56 (which together form the conventional COMMAND.COM DOS program) comprise the conventional DOS kernel. IBMBIO.COM provides low level device support. IBMDOS.COM provides application support. COMMAND.COM is the command processor. COMMAND 54 stays resident in RAM 14 once it is loaded while COMMAND 56 can be overlaid and later copied into RAM 14 as needed. ROMSHELL 57 provides a graphical user interface and is transient while SHELSTUB 59 is designed to be resident in the RAM for reloading the ROMSHELL program after quitting an application. The ROM DOS 34 programs when loaded into the RAM generally operate the same as the corresponding programs in DISK DOS 36, the only difference being that such programs are loaded from ROM 16 instead of disk 24. Relative to other portions of DOS and the system, ROM DOS 34 is transparent and appears to act the same as corresponding portions of DISK DOS 36. The ROM DOS files are stored in ROM 16 outside of the address space available for DOS programs, and RAM LOADER 51 is able to work outside such address space to copy such programs into the DOS address space in RAM 14. RAM LOADER 51 uses a standard method of moving blocks of data from the protected mode address space into the real mode DOS address space, such as by using the well known system service interrupt 15H of BIOS 40 which is invoked by setting the well known register AH (not shown) of microprocessor 12 to the MOVE BLOCK function 87H. Such function transfers a block of data from storage above the IMB protected mode address

> range by switching to the protected mode. RAM LOADER 51 is invoked using interrupt 2BH and accesses the information from items 42 and 46 to load the ROM header at initialization time, check for the existence of a module in ROM, load a module from ROM, and access flag 58 in CMOS RAM 20.

Feigenbaum, 6:3-44

> RAM LOADER 51, IFS HANDLER 53, and CHECKC.COM 63 are new with the invention, while the remaining programs stored in ROM 16 perform functions previously commercially available. ROM 16 may optionally be of a larger size to store additional programs such as a code page switching program, a program providing extended keyboard layout, support for other national languages, etc. It may also contain alternative AUTOEXEC.BAT and CONFIG.SYS files 102 and 104 oriented to the use of such additional programs. The size of ROM 16 may thus vary dependent upon the size of and how many additional support programs are stored. Preferably, those items shown in FIG. 3 are stored in a 128K ROM unit and the additional programs are stored in a second 128 K ROM unit.

Feigenbaum, 6:45-59

> ROMBOOT 65 first loads IBMBIO.COM 50, including RAM LOADER 51, from ROM 16 into RAM 14 by step 76. IBMBIO.COM includes two code segments, IBMINIT and SYSINIT. Step 78 then transfers control to IBMINIT 66. Step 80 initializes the system device drivers and hardware. Step 81 provides or sets up a hook into interrupt handler INT 2BH for the RAM LOADER. Step 82 relocates the initialization routine of IBMBIO.COM 50 to predetermined locations at the top of the DOS address space in RAM 16. Next, step 84 transfers control to SYSINIT 67 which first, by step 86, loads IBMDOS.COM 52 from ROM 16 into RAM 14. Step 88 then executes the initialization routine of IBMDOS.COM. Afterwards, step 90 attaches the IFS Handler as the next available drive which is drive D: because system 10 includes a fixed disk drive. Next, step 91 selects the ROM drive created by the IFS handler, as the defualt drive.

Feigenbaum, 8:53-9:2

> In summary, the invention of a ROM based DOS provides several features and advantages. DOS is rapidly loaded from ROM and executed in RAM to instantly boot up and present the user with a screen from a graphical user interface. Once booted, ROM DOS executes normally and acts the same as standard disk DOS and has the same system compatibility as disk DOS. Further, no DOS installation is required to be done by a user, and the user has no diskette dependencies. The invention is further advantageous by virtue of using the IFS interface and allowing the ROM to appear as a "drive", since this

minimizes the number of changes to the basic DOS programs, which reduced developement time and minimized the possibility of error being introduced by new code. Feigenbaum, 10:18-33

19.2 associating the accessed boot data that is not associated with the boot data list to the boot data list.	Feigenbaum, as evidenced by the example citations below, discloses "associating the accessed boot data that is not associated with the boot data list to the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, associating the accessed boot data that is not associated with the boot data list to the boot data list.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 2 above.	

23. The method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files.	Feigenbaum, as evidenced by the example citations below, discloses “the method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 16 above.	

Appendix C4
Invalidity of U.S. Patent 8,880,862 based on Feigenbaum

24. The method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system.
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 15 above.	

27. The method of claim 1, wherein the memory comprises: a physical memory.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 1, wherein the memory comprises: a physical memory."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

> A data processing system, such as a personal computer, contains bootable DOS programs that are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation.

Feigenbaum, Abstract

> When such personal computers are powered up, a power on self test (POST) routine or program is first executed, such program being stored in ROM. Upon the successful completion of such test, portions of DOS are read into the system memory from a hard disk or a floppy diskette and the system is booted up and initialized. The test, booting and initialization process can take many seconds. The present invention is directed to an improvement by which such process is significantly shortened in time. It is also common to start a personal computer by turning on not only a system unit but also a display, and the invention has an objective of accomplishing the start up process so that the system is available for use within the time required to warm up the display. That is, as soon as the system is turned on, the first screen becomes immediately visible on a display and gives the user the appearance of an "instant" load and initialization.

Feigenbaum, 1:22-39

> It is also known that the speed at which a ROM can be accessed is several orders of magnitude faster than the speed at which a hard disk or a floppy diskette can be accessed. The invention takes advantage of this known speed difference by storing portions of DOS in a ROM where such portions can be accessed at a speed much faster than the speed at which DOS could be accessed if such portions of DOS were stored on a hard disk or floppy diskette. However, the invention is more than simply substituting a ROM for a disk because of several problems. A first thought that might occur to many persons is that by storing DOS in a ROM, DOS can then be executed directly from the ROM.

> However, while certain portions of DOS, such as commands in the COMMAND.COM program, can be executed directly from ROM, other portions, such as IBMBIO.COM and IBMDOS.COM are altered and cannot be used in a read only device which precludes any alteration.

Feigenbaum, 1:40-57

> System 10 is further provided with a DOS 32 that comprises two major portions, ROM DOS 34 and DISK DOS 36 respectively stored in ROM 16 and disk 24. ROM DOS 34 includes the programs that are booted into the RAM when the system is initially powered on or when the system is reset, which programs provide the minimum level of operating system support to make system 10 operational to the user. The remaining portions of DOS, i.e. those programs and files that together form a complete DOS, are included in DISK DOS 36. DISK DOS 36 further includes DOS programs similar to those in ROM DOS so that the system can be booted up and operated from disk 24 at the option of the user, as described in more detail hereafter. Also stored in ROM 16 are a basic input/output system (BIOS) 38 and a power on self test (POST) routine 40. When system 10 is first turned on, or when it is restarted, POST 40 first performs the test and upon successful completion, it boots or loads a portion of DOS and transfers control to it in the manner described in detail below. Except for ROM DOS 34, the system as thus far described is constructed and operates in accordance with known principles of the prior ar

Feigenbaum, 5:3-25

> In summary, the invention of a ROM based DOS provides several features and advantages. DOS is rapidly loaded from ROM and executed in RAM to instantly boot up and present the user with a screen from a graphical user interface. Once booted, ROM DOS executes normally and acts the same as standard disk DOS and has the same system compatibility as disk DOS. Further, no DOS installation is required to be done by a user, and the user has no diskette dependencies. The invention is further advantageous by virtue of using the IFS interface and allowing the ROM to appear as a "drive", since this minimizes the number of changes to the basic DOS programs, which reduced developement time and minimized the possibility of error being introduced by new code.

Feigenbaum, 10:18-33

28. The method of claim 1, wherein the operating system comprises: a plurality of files.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 1, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 16 above.	

29. The method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 15 and 17 above.	

31. The method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

> It is also known that the speed at which a ROM can be accessed is several orders of magnitude faster than the speed at which a hard disk or a floppy diskette can be accessed. The invention takes advantage of this known speed difference by storing portions of DOS in a ROM where such portions can be accessed at a speed much faster than the speed at which DOS could be accessed if such portions of DOS were stored on a hard disk or floppy diskette. However, the invention is more than simply substituting a ROM for a disk because of several problems. A first thought that might occur to many persons is that by storing DOS in a ROM, DOS can then be executed directly from the ROM. However, while certain portions of DOS, such as commands in the COMMAND.COM program, can be executed directly from ROM, other portions, such as IBMBIO.COM and IBMDOS.COM are altered and cannot be used in a read only device which precludes any alteration.

Feigenbaum, 1:40-57

> FIG. 4 illustrates the relative position and use of IFS HANDLER 53. Within the prior art, an application program 110 can be thought of as sitting on top of FAT file system 112, which in turn sits on top of disk/diskette driver 114 which overlies the disk 24 containing the actual files and data. Application 110 uses interrupt INT 21H to access the file system through IBMDOS.COM, which acts through IBMBIO.COM to access the disk. In accordance with the invention, IFS HANDLER 53 is interposed at the file system level in the INT 21H process and decides in step 116 whether the desired file system access is to be made to ROM DISK D:. If not, the system proceeds as in the prior art to access the FAT file system. If the desired access is to be made to ROM DISK D:, it is done through the ROM file system 118 and RAM LOADER 51 by programs IBMDOS.COM 52 and IBMBIO.COM 50. The term "ROM DISK" is defined hereby to mean a ROM unit containing images of files storable on a hard disk

or floppy diskette, which images are stored in a packed format in a ROM file system, that is different from the FAT file system used to store files on a disk or diskette.

Feigenbaum, 9:39-60

Programs 50 through 56 provide the minimum level of operating system support for operating data processing system 10. IBMBIO.COM 50 (except for RAM LOADER 51), IBMDOS.COM 52 (except for IFS handler 53), and COMMANDS 54 AND 56 (which together form the conventional COMMAND.COM DOS program) comprise the conventional DOS kernel. IBMBIO.COM provides low level device support. IBMDOS.COM provides application support. COMMAND.COM is the command processor. COMMAND 54 stays resident in RAM 14 once it is loaded while COMMAND 56 can be overlaid and later copied into RAM 14 as needed. ROMSHELL 57 provides a graphical user interface and is transient while SHELSTUB 59 is designed to be resident in the RAM for reloading the ROMSHELL program after quitting an application. The ROM DOS 34 programs when loaded into the RAM generally operate the same as the corresponding programs in DISK DOS 36, the only difference being that such programs are loaded from ROM 16 instead of disk 24. Relative to other portions of DOS and the system, ROM DOS 34 is transparent and appears to act the same as corresponding portions of DISK DOS 36. The ROM DOS files are stored in ROM 16 outside of the address space available for DOS programs, and RAM LOADER 51 is able to work outside such address space to copy such programs into the DOS address space in RAM 14. RAM LOADER 51 uses a standard method of moving blocks of data from the protected mode address space into the real mode DOS address space, such as by using the well known system service interrupt 15H of BIOS 40 which is invoked by setting the well known register AH (not shown) of microprocessor 12 to the MOVE BLOCK function 87H. Such function transfers a block of data from storage above the IMB protected mode address range by switching to the protected mode. RAM LOADER 51 is invoked using interrupt 2BH and accesses the information from items 42 and 46 to load the ROM header at initialization time, check for the existence of a module in ROM, load a module from ROM, and access flag 58 in CMOS RAM 20.

Feigenbaum, 6:3-45

Another object is to store a DOS kernel in a ROM in a simple, efficient file structure that can be rapidly accessed using less code than that required to access a DOS kernel stored on a disk in a FAT file system.

Feigenbaum, 3:34-37

A still further object is to provide improved method and apparatus for rapidly

> initializing a personal computer when it is turned on, which method and apparatus uses an IFS to access DOS programs stored in a ROM in a packed format differing from the format used to store such programs in a FAT file system.

Feigenbaum, 3:47-52

> The IFS interface "connects" the ROM as an installable file system and assigns it a drive letter to be referenced. Such interface performs calls at a file system level as opposed to other systems which operate at the disk level. Examples of interfaces operating at the system level are OPEN, READ, CLOSE, etc. Examples of interfaces which operate at the disk level are READ SECTOR, and WRITE SECTOR. Operating system performance is improved since file system calls are identified earlier in the code path than disk calls. Further, in view of the relative simplicity of the ROM file structure in comparison to the complex file structure of the FAT file system, it is obvious that the amount of code necessary to access the ROM file system is substantially less than the amount of code necessary to access the FAT file system. Thus performance is faster because less code has to be executed and because the FAT file system code is bypassed due to the system level interrupt being located in the code path prior to the FAT file system code in IBMDOS.COM.

Feigenbaum, 9:61-10:12

Appendix C4
Invalidity of U.S. Patent 8,880,862 based on Feigenbaum

32. The method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

33. The method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 32 above.	

35. The method of claim 5, wherein the compressed boot data represents a plurality of files.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 5, wherein the compressed boot data represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the compressed boot data represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 16 and 23 above.	

Appendix C4
Invalidity of U.S. Patent 8,880,862 based on Feigenbaum

36. The method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system.	Feigenbaum, as evidenced by the example citations below, discloses “the method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 15 and 17 above.	

39. The method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory.	Feigenbaum, as evidenced by the example citations below, discloses “the method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory”

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.

Feigenbaum discloses this limitation:

> A data processing system, such as a personal computer, contains bootable DOS programs that are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation.

Feigenbaum, Abstract

> When such personal computers are powered up, a power on self test (POST) routine or program is first executed, such program being stored in ROM. Upon the successful completion of such test, portions of DOS are read into the system memory from a hard disk or a floppy diskette and the system is booted up and initialized. The test, booting and initialization process can take many seconds. The present invention is directed to an improvement by which such process is significantly shortened in time. It is also common to start a personal computer by turning on not only a system unit but also a display, and the invention has an objective of accomplishing the start up process so that the system is available for use within the time required to warm up the display. That is, as soon as the system is turned on, the first screen becomes immediately visible on a display and gives the user the appearance of an "instant" load and initialization.

Feigenbaum, 1:22-39

> It is also known that the speed at which a ROM can be accessed is several orders of magnitude faster than the speed at which a hard disk or a floppy

> diskette can be accessed. The invention takes advantage of this known speed difference by storing portions of DOS in a ROM where such portions can be accessed at a speed much faster than the speed at which DOS could be accessed if such portions of DOS were stored on a hard disk or floppy diskette. However, the invention is more than simply substituting a ROM for a disk because of several problems. A first thought that might occur to many persons is that by storing DOS in a ROM, DOS can then be executed directly from the ROM. However, while certain portions of DOS, such as commands in the COMMAND.COM program, can be executed directly from ROM, other portions, such as IBMBIO.COM and IBMDOS.COM are altered and cannot be used in a read only device which precludes any alteration.

Feigenbaum, 1:40-57

> System 10 is further provided with a DOS 32 that comprises two major portions, ROM DOS 34 and DISK DOS 36 respectively stored in ROM 16 and disk 24. ROM DOS 34 includes the programs that are booted into the RAM when the system is initially powered on or when the system is reset, which programs provide the minimum level of operating system support to make system 10 operational to the user. The remaining portions of DOS, i.e. those programs and files that together form a complete DOS, are included in DISK DOS 36. DISK DOS 36 further includes DOS programs similar to those in ROM DOS so that the system can be booted up and operated from disk 24 at the option of the user, as described in more detail hereafter. Also stored in ROM 16 are a basic input/output system (BIOS) 38 and a power on self test (POST) routine 40. When system 10 is first turned on, or when it is restarted, POST 40 first performs the test and upon successful completion, it boots or loads a portion of DOS and transfers control to it in the manner described in detail below. Except for ROM DOS 34, the system as thus far described is constructed and operates in accordance with known principles of the prior art.

Feigenbaum, 5:3-25

> In summary, the invention of a ROM based DOS provides several features and advantages. DOS is rapidly loaded from ROM and executed in RAM to instantly boot up and present the user with a screen from a graphical user interface. Once booted, ROM DOS executes normally and acts the same as standard disk DOS and has the same system compatibility as disk DOS. Further, no DOS installation is required to be done by a user, and the user has no diskette dependencies. The invention is further advantageous by virtue of using the IFS interface and allowing the ROM to appear as a "drive", since this minimizes the number of changes to the basic DOS programs, which reduced developement time and minimized the possibility of error being introduced by new code.

Feigenbaum, 10:18-33

40. The method of claim 36, wherein the operating system comprises: a plurality of files.	Feigenbaum, as evidenced by the example citations below, discloses “the method of claim 36, wherein the operating system comprises: a plurality of files”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 36, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 16 and 23 above.	

43. The method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 31 above.	

44. The method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 32 above.	

45. The method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 32 and 33 above.	

47. The system of claim 6, wherein the boot data in the compressed form represents a plurality of files.	Feigenbaum, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 16 and 23 above.	

48. The system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system.	Feigenbaum, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 15, 17 and 24 above.	

49. The system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form.	Feigenbaum, as evidenced by the example citations below, discloses “the system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form”

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.

Feigenbaum discloses this limitation:

See Claim 10 above.

> A data processing system, such as a personal computer, contains bootable DOS programs that are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation.

Feigenbaum, Abstract

> A still further object is to provide improved method and apparatus for rapidly initializing a personal computer when it is turned on, which method and apparatus uses an IFS to access DOS programs stored in a ROM in a packed format differing from the format used to store such programs in a FAT file system.

Feigenbaum, 3:47-52

> FIG. 3 also illustrates another facet of the invention which is that the data and programs of ROM DOS 34 are stored in ROM 16 in a file system using a packed format. In contrast, any data and files stored on disk 24, including those in RAM DOS 36, are stored in the conventional DOS FAT file system in which information is stored in clusters and sectors. In the FAT file system, if a particular program doesn't have the same number of bytes as are in a sector or cluster, space is wasted. On the average, about one half the number of bytes in a cluster or in a sector are wasted for each file. Since ROM units are more expensive than disk storage, any wasted space is inefficient. Thus, in ROM 16, the ROM DOS 34 files are stored beginning at a segment address, each segment being sixteen bytes. Each succeeding file begins immediately at the next

> segment location following the end of a preceding file so that on average only eight bytes per file would be wasted, such number being far less than the average waste in a FAT file system. By way of example, suppose file 50 begins at ROM segment X and ends at Y. Then, the succeeding file 52 begins at segment Y+n, where n is less than sixteen. The other files are similarly stored.

Feigenbaum, 6:60-7:15

> Briefly, in accordance with the subject invention, bootable DOS programs are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation.

Feigenbaum, 3:53-59

> FIG. 4 illustrates the relative position and use of IFS HANDLER 53. Within the prior art, an application program 110 can be thought of as sitting on top of FAT file system 112, which in turn sits on top of disk/diskette driver 114 which overlies the disk 24 containing the actual files and data. Application 110 uses interrupt INT 21H to access the file system through IBMDOS.COM, which acts through IBMBIO.COM to access the disk. In accordance with the invention, IFS HANDLER 53 is interposed at the file system level in the INT 21H process and decides in step 116 whether the desired file system access is to be made to ROM DISK D:. If not, the system proceeds as in the prior art to access the FAT file system. If the desired access is to be made to ROM DISK D:, it is done through the ROM file system 118 and RAM LOADER 51 by programs IBMDOS.COM 52 and IBMBIO.COM 50. The term "ROM DISK" is defined hereby to mean a ROM unit containing images of files storable on a hard disk or floppy diskette, which images are stored in a packed format in a ROM file system, that is different from the FAT file system used to store files on a disk or diskette.

Feigenbaum, 9:39-60

50. The system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form.	Feigenbaum, as evidenced by the example citations below, discloses “the system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.	

51. The system of claim 6, wherein the first memory comprises: a physical memory.	Feigenbaum, as evidenced by the example citations below, discloses "the system of claim 6, wherein the first memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the first memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 27 and 39 above.	

52. The system of claim 6, wherein the boot data the compressed form comprises: a plurality of files.	Feigenbaum, as evidenced by the example citations below, discloses “the system of claim 6, wherein the boot data the compressed form comprises: a plurality of files”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data the compressed form comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 16 and 23 above.	

53. The system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program.	Feigenbaum, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 15, 17 and 24 above.	

59. The method of claim 8, wherein the operating system in the compressed form represents a plurality of files.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 16 and 23 above.	

60. The method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 15, 17 and 24 above.	

63. The method of claim 8, wherein the second memory comprises: a physical memory.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 8, wherein the second memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the second memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 27 and 38 above.	

Appendix C4
Invalidity of U.S. Patent 8,880,862 based on Feigenbaum

64. The method of claim 8, wherein the operating system comprises: a plurality of files.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 16 and 23 above.	

65. The method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program.	Feigenbaum, as evidenced by the example citations below, discloses “the method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 15, 17 and 24 above.	

67. The method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 31 above.	

71. The method of claim 11, wherein the boot data in the compressed form represents a plurality of files.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 11, wherein the boot data in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 16 and 23 above.	

72. The method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 15, 17 and 24 above.	

75. The method of claim 11, wherein the memory comprises: a physical memory.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 11, wherein the memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 27 above.	

76. The method of claim 11, wherein the operating system comprises: a plurality of files.	Feigenbaum, as evidenced by the example citations below, discloses “the method of claim 11, wherein the operating system comprises: a plurality of files”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 16 and 23 above.	

77. The method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 15, 17 and 24 above.	

79. The method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 31 above.	

80. The method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 32, 33 and 49 above.	

Appendix C4
Invalidity of U.S. Patent 8,880,862 based on Feigenbaum

81. The method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 32, 33 and 49 above.	

83. The method of claim 13, wherein the boot data in the compressed form represents a plurality of files.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 13, wherein the boot data in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 16 and 23 above.	

84. The method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 15, 17 and 24 above.	

87. The method of claim 13, wherein the memory comprises: a physical memory.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 13, wherein the memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 27 above.	

88. The method of claim 13, wherein the operating system comprises: a plurality of files.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 13, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 16 and 23 above.	

89. The method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 15, 17 and 24 above.	

91. The method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 31 above.	

92. The method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 32, 33 and 49 above.	

93. The method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claim 32, 33, and 49 above.	

97.1 The method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list,	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 9.1-9.3 and 19 above.	

97.2 and wherein the updating comprises: associating the additional compressed boot data with the boot data list.	Feigenbaum, as evidenced by the example citations below, discloses "and wherein the updating comprises: associating the additional compressed boot data with the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, and wherein the updating comprises: associating the additional compressed boot data with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 1.4.1, 2, 4, 5.6, and 8.6 above.	

98. The method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list.	Feigenbaum, as evidenced by the example citations below, discloses “the method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 1.4.1, 5.6, and 8.6 above.	

107. The method of claim 2, further comprising: storing the updated boot list in a non-volatile memory.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 2, further comprising: storing the updated boot list in a non-volatile memory."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 2, further comprising: storing the updated boot list in a non-volatile memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Feigenbaum discloses this limitation:

See Claims 1.4.1, 5.6, and 8.6 above.

> System 10 also includes a circuit or bus network 18 operatively interconnecting the various elements of the system. A complementary metal oxide semi-conductor (CMOS) RAM 20 is connected to and backed up by a battery 22 to provide non-volatile storage. A disk drive 26 includes a fixed disk 24 for storing information in a FAT file system. Drive 26 and CMOS RAM 20 are also connected to bus 18.

Feigenbaum, 4:54-5:2

108. The method of claim 2, further comprising: compressing at least a portion of the additional boot data.	Feigenbaum, as evidenced by the example citations below, discloses “the method of claim 2, further comprising: compressing at least a portion of the additional boot data”

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 2, further comprising: compressing at least a portion of the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.

Feigenbaum discloses this limitation:

See Claims 1.1, 5, 9.1 and 8 above.

> A data processing system, such as a personal computer, contains bootable DOS programs that are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation.

Feigenbaum, Abstract

> Briefly, in accordance with the subject invention, bootable DOS programs are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation.

Feigenbaum, 3:53-59

109. The method of claim 108, further comprising: storing the compressed additional boot data.	Feigenbaum, as evidenced by the example citations below, discloses "the method of claim 108, further comprising: storing the compressed additional boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 108, further comprising: storing the compressed additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Feigenbaum discloses this limitation: *See* Claims 5.1, 8.1 and 9.1 above. > A data processing system, such as a personal computer, contains bootable DOS programs that are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation. Feigenbaum, Abstract > Briefly, in accordance with the subject invention, bootable DOS programs are stored in a ROM as an alternate file system in which the files are stored in packed format. When the system is powered on, the programs are rapidly booted up or loaded from ROM into RAM and executed to "instantly" (as it appears to the user) place the system in operation. Feigenbaum, 3:53-59	

Appendix C5
Invalidity of U.S. Patent 8,880,862 based on Greene

U.S. Patent No. 5,836,013 to Greene ("Greene") invalidates claims 1-6, 8-11, 13-17, 19, 23-24, 27-29, 31-33, 35-36, 39-40, 43-45, 47-53, 59-60, 63-65, 67, 71-72, 75-77, 79-81, 83-84, 87-89, 91-93, 97-98, and 107-109 of United States Patent No. 8,880,862 ("the '862 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

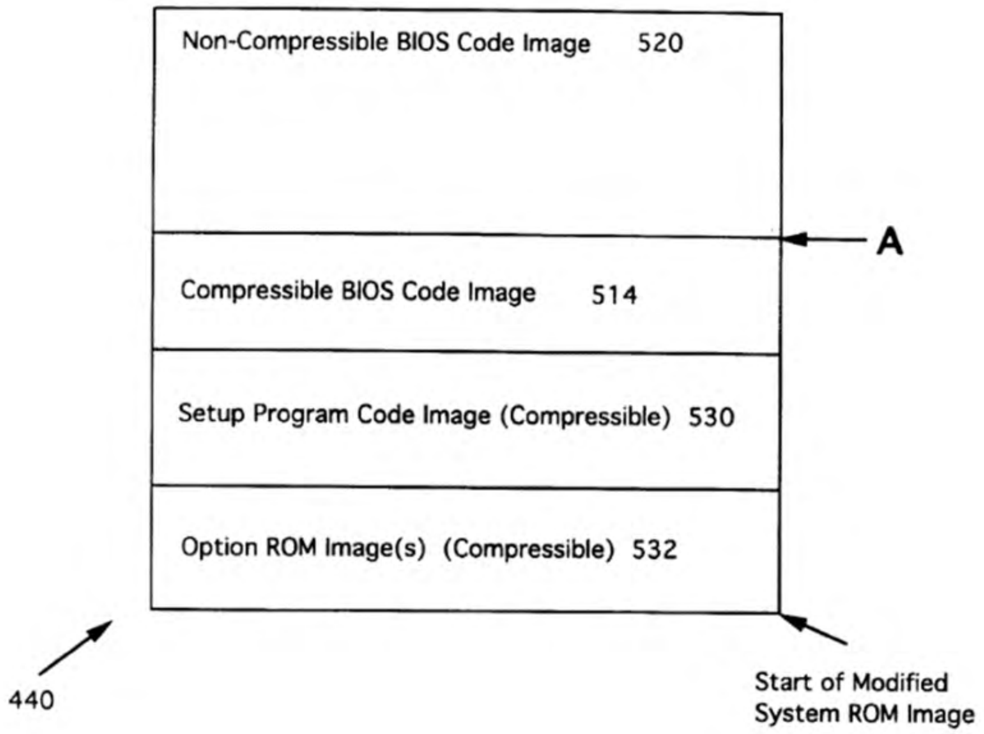
The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '862 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix C5
Invalidity of U.S. Patent 8,880,862 based on Greene

1 (Preamble) A method for providing accelerated loading of an operating system in a computer system, the method comprising:	Greene, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, the method comprising:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system in a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Greene discloses this limitation:



Non-Compressible BIOS Code Image 520
A
Compressible BIOS Code Image 514
Setup Program Code Image (Compressible) 530
Option ROM Image(s) (Compressible) 532
440
Start of Modified System ROM Image

Greene, Fig. 5.

> "A chipset (platform)-independent method and apparatus for compressing and decompressing a system ROM of a computer (e.g., BIOS, setup program, and one or more option ROMs) are disclosed. The setup program, option ROM, and part of the BIOS are compressed using a lossless compression algorithm. A non-compressible portion of the BIOS includes a decompression algorithm and a shadow RAM block table of chipset-specific registers and bit patterns to write-enable

> and read-enable shadow RAM (RAM that is mapped to the ROM address space). The compressed data is stored in a compressed data block format with the associated location in memory to decompress the compressed data."

Greene, Abstract. *See also* 1:40-1:63.

> "During the BIOS Power-On Self-Test (POST) process, the compressed system ROM is copied to conventional memory, and the decompression program is executed."

Greene, Abstract. *See also* 1:64-2:13.

> "System ROM 210 comprises BIOS (Basic Input/Output System) 310, and optionally setup program 320, and one or more option ROM images for input/output (I/O) devices associated with the computer 332. An "image" is a binary representation of code and/or data of a computer file. BIOS 310 comprises a plurality of routines that initialize the computer hardware and provide primitive I/O services that the operating system and application programs use to manipulate hardware associated with the computer. System setup program 320 is a utility that allows the end user to configure BIOS 310."

Greene, 3:16-28, Fig. 3.

1.1 loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory;	Greene, as evidenced by the example citations below, discloses “loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory;”

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.

Greene discloses this limitation:

> “The compressed data is stored in a compressed data block format with the associated location in memory to decompress the compressed data. Thus, the data can be decompressed anywhere in memory of a target computer. For example, the BIOS is decompressed to shadow RAM and the setup program is decompressed to conventional memory.”

Greene, Abstract.

> “The decompression program scans the compressed system ROM for compressed data blocks and decompresses the compressed data therein to the associated locations in memory. If compressed data is located in shadow RAM, shadow RAM is enabled for writing and reading with reference to the chipset-specific information in the shadow RAM block table. If compressed data was decompressed to conventional memory space, this space is cleared before exiting the POST process.”

Greene, Abstract.

> “Compressed data is stored in a compressed data block comprising the compressed data and various associated information including the location in memory to place the compressed data when decompressed. The compressed system ROM is stored in ROM of a target computer.”

Greene, 1:58-63.

> "During the BIOS Power-On Self-Test (POST) process of the target computer, the compressed system ROM image is copied to conventional memory (e.g., RAM), and the decompression program is executed. The decompression program write-enables shadow RAM (with reference to the chipset-specific information in the shadow RAM block table), copies the non-compressible BIOS code image from conventional memory to shadow RAM, and read-enable shadow RAM. The decompression program scans the compressed system ROM image for compressed data blocks and decompresses the compressed data therein to the associated locations in memory."

Greene, 1:64-2:8.

> "In a preferred embodiment, the destination location in memory 100 (e.g., address space) of decompressed data is specified in decompressed data start 722. Thus, compressed data 702 can be placed anywhere in memory 100 upon decompression."

Greene, 5:53-6:2

> "Compressed system ROM image 632 is copied 802 from ROM 130 to conventional memory 110, (e.g., RAM 112)."

Greene, 7:27-29, Fig. 8a.

1.2 accessing the loaded portion of the boot data in the compressed form from memory;	Greene, as evidenced by the example citations below, discloses "accessing the loaded portion of the boot data in the compressed form from memory;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded portion of the boot data in the compressed form from memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Greene discloses this limitation:

> "The compressed data is stored in a compressed data block format with the associated location in memory to decompress the compressed data. Thus, the data can be decompressed anywhere in memory of a target computer. For example, the BIOS is decompressed to shadow RAM and the setup program is decompressed to conventional memory."

Greene, Abstract.

> "The decompression program scans the compressed system ROM for compressed data blocks and decompresses the compressed data therein to the associated locations in memory. If compressed data is located in shadow RAM, shadow RAM is enabled for writing and reading with reference to the chipset-specific information in the shadow RAM block table. If compressed data was decompressed to conventional memory space, this space is cleared before exiting the POST process."

Greene, Abstract.

> "During the BIOS Power-On Self-Test (POST) process of the target computer, the compressed system ROM image is copied to conventional memory (e.g., RAM), and the decompression program is executed. The decompression program write-enables shadow RAM (with reference to the the chipset-specific information in the shadow RAM block table), copies the non-compressible BIOS code image from conventional memory to shadow RAM, and read-enables shadow RAM. The decompression program scans the compressed system ROM image for compressed data blocks and decompresses the compressed data therein to the associated locations in memory. If data is located in shadow RAM, shadow RAM is write-enabled and read-enabled (with reference to the chipset-specific information in the shadow RAM block table). If

> compressed data was decompressed to conventional memory, this space is cleared before exiting the POST process."

Greene, 1:64-2:13.

1.3 decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form; and	Greene, as evidenced by the example citations below, discloses "decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Greene discloses this limitation:

> "The compressed data is stored in a compressed data block format with the associated location in memory to decompress the compressed data. Thus, the data can be decompressed anywhere in memory of a target computer. For example, the BIOS is decompressed to shadow RAM and the setup program is decompressed to conventional memory."

Greene, Abstract.

> "The decompression program scans the compressed system ROM for compressed data blocks and decompresses the compressed data therein to the associated locations in memory. If compressed data is located in shadow RAM, shadow RAM is enabled for writing and reading with reference to the chipset-specific information in the shadow RAM block table. If compressed data was decompressed to conventional memory space, this space is cleared before exiting the POST process."

Greene, Abstract.

> "During the BIOS Power-On Self-Test (POST) process of the target computer, the compressed system ROM image is copied to conventional memory (e.g., RAM), and the decompression program is executed. The decompression program write-enables shadow RAM (with reference to the the chipset-specific information in the shadow RAM block table), copies the non-compressible BIOS code image from conventional memory to shadow RAM, and read-enables shadow RAM. The decompression program scans the compressed system ROM image for

> compressed data blocks and decompresses the compressed data therein to the associated locations in memory. If data is located in shadow RAM, shadow RAM is write-enabled and read-enabled (with reference to the chipset-specific information in the shadow RAM block table). If compressed data was decompressed to conventional memory, this space is cleared before exiting the POST process."

Greene, 1:64-2:13.

> "Referring to FIG. 10, decompression program 422 comprises a decompression algorithm 1102. Decompression algorithm 1102 corresponds to the compression algorithm used (e.g., LZSS or LZARI)."

Greene, 3:66-4:2.

> "Compressed system ROM 632 image must be decompressed before the compressed data therein can be used on the target computer. In a preferred embodiment, decompression is done early in the POST process."

Greene, 7:21-24.

> "Program execution control is transferred 804 to decompression program 422 (in non-compressible BIOS code image 520 of compressed system ROM image 632)."

Greene, 7:29-32, Fig. 8a. *See also* 7:33-7:64.

> "The decompression scan process repeats 832 until each compressed data block 700 in compressed system ROM image 632 is decompressed. In a preferred embodiment, program execution control is then transferred 834 to BIOS 310, now running in shadow RAM 202. In one embodiment, if data was decompressed to conventional (RAM) memory 110, conventional memory 110 is cleared 836 at the end of the POST process and before the operating system is booted (for compatibility reasons)."

Greene, 7:65-8:21.

1.4.1 updating the boot data list,	Greene, as evidenced by the example citations below, discloses "updating the boot data list,"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

1.4.2 wherein the decompressed portion of boot data comprises a portion of the operating system.	Greene, as evidenced by the example citations below, discloses "wherein the decompressed portion of boot data comprises a portion of the operating system."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the decompressed portion of boot data comprises a portion of the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Greene discloses this limitation:

Greene discloses this claim limitation.

> "During the BIOS Power-On Self-Test (POST) process of the target computer, the compressed system ROM image is copied to conventional memory (e.g., RAM), and the decompression program is executed. The decompression program write-enables shadow RAM (with reference to the the chipset-specific information in the shadow RAM block table), copies the non-compressible BIOS code image from conventional memory to shadow RAM, and read-enables shadow RAM. The decompression program scans the compressed system ROM image for compressed data blocks and decompresses the compressed data therein to the associated locations in memory. If data is located in shadow RAM, shadow RAM is write-enabled and read-enabled (with reference to the chipset-specific information in the shadow RAM block table). If compressed data was decompressed to conventional memory, this space is cleared before exiting the POST process."

Greene, 1:64-2:13.

> "Referring to FIG. 10, decompression program 422 comprises a decompression algorithm 1102. Decompression algorithm 1102 corresponds to the compression algorithm used (e.g., LZSS or LZARI)."

Greene, 3:66-4:2.

> "Compressed system ROM 632 image must be decompressed before the compressed data therein can be used on the target computer. In a preferred embodiment, decompression is done early in the POST process."

Greene, 7:21-24.

> "Program execution control is transferred 804 to decompression program 422 (in non-compressible BIOS code image 520 of compressed system ROM image 632)."

Greene, 7:29-32, Fig. 8a. *See also* 7:33-7:64.

> "The decompression scan process repeats 832 until each compressed data block 700 in compressed system ROM image 632 is decompressed. In a preferred embodiment, program execution control is then transferred 834 to BIOS 310, now running in shadow RAM 202. In one embodiment, if data was decompressed to conventional (RAM) memory 110, conventional memory 110 is cleared 836 at the end of the POST process and before the operating system is booted (for compatibility reasons)."

Greene, 7:65-8:21.

2. The method of claim 1, wherein the updating comprises: associating additional boot data with the boot data list.	Greene, as evidenced by the example citations below, discloses "wherein the updating comprises: associating additional boot data with the boot data list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein associating additional boot data with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.4.1 above.	

3. The method of claim 1, wherein the updating comprises: removing an association of additional boot data that is associated with the boot data list from the boot data list.	Greene, as evidenced by the example citations below, discloses "wherein the updating comprises: removing an association of additional boot data that is associated with the boot data list from the boot data list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein removing an association of additional boot data that is associated with the boot data list from the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.4.1 above.	

4. The method of claim 1, wherein the updating comprises: associating additional boot data with the boot data list; and compressing a portion of the additional boot data.	Greene, as evidenced by the example citations below, discloses "wherein the updating comprises: associating additional boot data with the boot data list; and compressing a portion of the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein associating additional boot data with the boot data list; and compressing a portion of the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 1.4.1, and 2 above.	

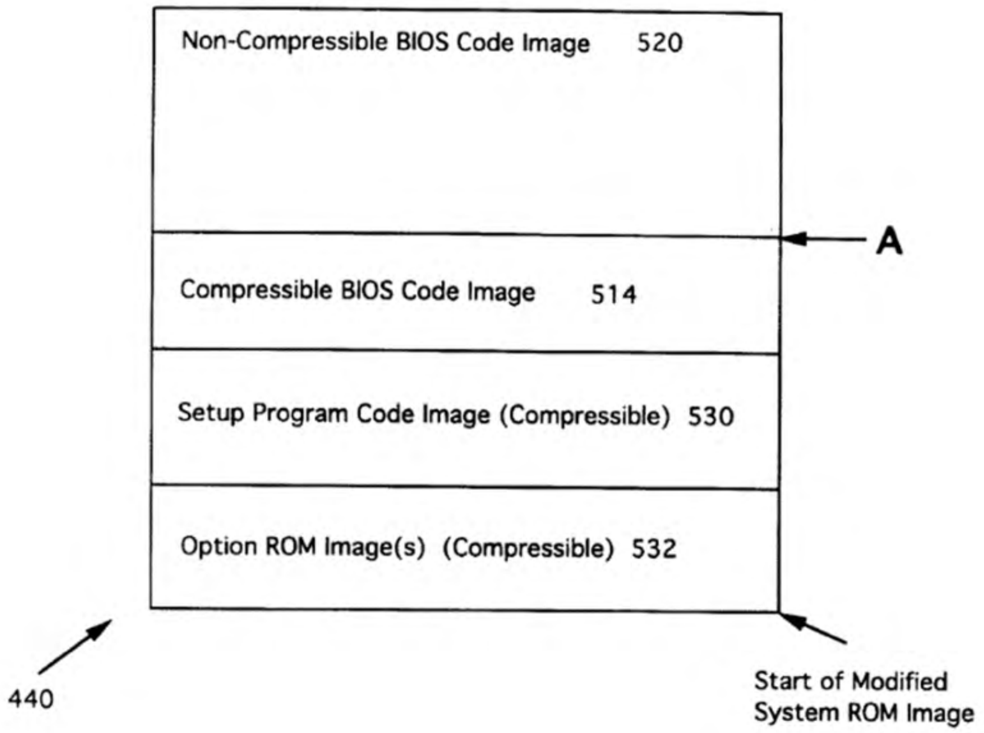
5 (Preamble) A method for booting a computer system, the method comprising:	Greene, as evidenced by the example citations below, discloses "a method for booting a computer system, the method comprising:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Greene discloses this limitation:

See Claims 1-1.4.2 above.

See also **Add disclosure from '608 Patent Claims 1.1 and 1.2.**


Non-Compressible BIOS Code Image 520
A
Compressible BIOS Code Image 514
Setup Program Code Image (Compressible) 530
Option ROM Image(s) (Compressible) 532
440
Start of Modified System ROM Image

Greene, Fig. 5.

> "A chipset (platform)-independent method and apparatus for compressing and decompressing a system ROM of a computer (e.g., BIOS, setup program, and one or more option ROMs) are disclosed. The setup program, option ROM, and part of the BIOS are compressed using a lossless compression algorithm. A non-compressible portion of

the BIOS includes a decompression algorithm and a shadow RAM block table of chipset-specific registers and bit patterns to write-enable and read-enable shadow RAM (RAM that is mapped to the ROM address space). The compressed data is stored in a compressed data block format with the associated location in memory to decompress the compressed data."

Greene, Abstract. *See also* 1:40-1:63.

"During the BIOS Power-On Self-Test (POST) process, the compressed system ROM is copied to conventional memory, and the decompression program is executed."

Greene, Abstract. *See also* 1:64-2:13.

"System ROM 210 comprises BIOS (Basic Input/Output System) 310, and optionally setup program 320, and one or more option ROM images for input/output (I/O) devices associated with the computer 332. An "image" is a binary representation of code and/or data of a computer file. BIOS 310 comprises a plurality of routines that initialize the computer hardware and provide primitive I/O services that the operating system and application programs use to manipulate hardware associated with the computer. System setup program 320 is a utility that allows the end user to configure BIOS 310."

Greene, 3:16-28, Fig. 3.

"BIOS code 410, and optionally setup program code 430, and one or more option ROM code modules 432, are each developed, written (e.g., in any computer language such as C or Assembly language) and saved in a computer file. Code and data, as referred to herein, are used interchangeably and comprise computer code and/or data. BIOS code 410 is separated 412 into compressible BIOS code 414 and non-compressible BIOS code 420. Non-compressible BIOS code 420 comprises compression-related information table 424, decompression program 422, and shadow RAM block table 1104."

Greene, 3:42-52.

"Continuing with FIG. 4, compressible BIOS code 414, non-compressible BIOS code 420 (comprising decompression program 422 (1102, 1106, 1108), compression-related information table 424, and shadow RAM block table 1104), and optionally setup program code 430, and option ROM code module(s) 432, are compiled or assembled

and linked 434 to generate a modified system ROM image 440." Greene, 4:64-5:3.

5.1 storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory;	Greene, as evidenced by the example citations below, discloses "storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.1 above.	

5.2 loading the stored compressed boot data from the first memory;	Greene, as evidenced by the example citations below, discloses “loading the stored compressed boot data from the first memory”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the stored compressed boot data from the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.1 above.	

5.3 accessing the loaded compressed boot data;	Greene, as evidenced by the example citations below, discloses “accessing the loaded compressed boot data”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.2 above.	

5.4 decompressing the accessed compressed boot data;	Greene, as evidenced by the example citations below, discloses "decompressing the accessed compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.3 above.	

5.5 utilizing the decompressed boot data to at least partially boot the computer system;	Greene, as evidenced by the example citations below, discloses "utilizing the decompressed boot data to at least partially boot the computer system"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed boot data to at least partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Greene discloses this limitation:

See Claims 1.3 and 1.4.2 above.

See also

> "During the BIOS Power-On Self-Test (POST) process of the target computer, the compressed system ROM image is copied to conventional memory (e.g., RAM), and the decompression program is executed. The decompression program write-enables shadow RAM (with reference to the the chipset-specific information in the shadow RAM block table), copies the non-compressible BIOS code image from conventional memory to shadow RAM, and read-enables shadow RAM. The decompression program scans the compressed system ROM image for compressed data blocks and decompresses the compressed data therein to the associated locations in memory. If data is located in shadow RAM, shadow RAM is write-enabled and read-enabled (with reference to the chipset-specific information in the shadow RAM block table). If compressed data was decompressed to conventional memory, this space is cleared before exiting the POST process."

Greene, 1:64-2:13.

> "Referring to FIG. 10, decompression program 422 comprises a decompression algorithm 1102. Decompression algorithm 1102 corresponds to the compression algorithm used (e.g., LZSS or LZARI)."

Greene, 3:66-4:2.

> "Compressed system ROM 632 image must be decompressed before the compressed data therein can be used on the target computer. In a preferred embodiment, decompression is done early in the POST process."

Greene, 7:21-24.

> "Program execution control is transferred 804 to decompression program 422 (in non-compressible BIOS code image 520 of compressed system ROM image 632)."

Greene, 7:29-32, Fig. 8a. *See also* 7:33-7:64.

> "The decompression scan process repeats 832 until each compressed data block 700 in compressed system ROM image 632 is decompressed. In a preferred embodiment, program execution control is then transferred 834 to BIOS 310, now running in shadow RAM 202. In one embodiment, if data was decompressed to conventional (RAM) memory 110, conventional memory 110 is cleared 836 at the end of the POST process and before the operating system is booted (for compatibility reasons)."

Greene, 7:65-8:21.

5.6 updating the boot data list;	Greene, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.4.1 above.	

5.7 wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form.	Greene, as evidenced by the example citations below, discloses "wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 1-1.3 above.	

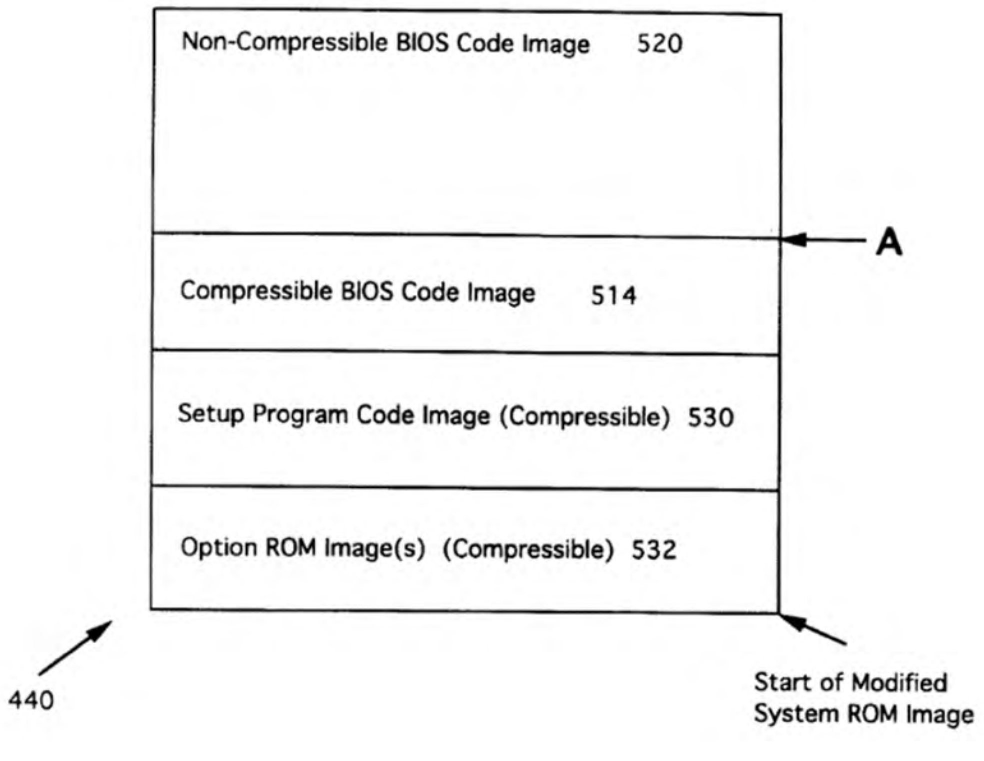
Appendix C5
Invalidity of U.S. Patent 8,880,862 based on Greene

6 (Preamble) A system comprising: a processor;	Greene, as evidenced by the example citations below, discloses "a system comprising: a processor:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system comprising: a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Greene discloses this limitation:

See ***Add disclosure from '608 Patent Claim 1.2***



Non-Compressible BIOS Code Image 520
A
Compressible BIOS Code Image 514
Setup Program Code Image (Compressible) 530
Option ROM Image(s) (Compressible) 532
440
Start of Modified System ROM Image

Greene, Fig. 5.

> "A chipset (platform)-independent method and apparatus for compressing and decompressing a system ROM of a computer (e.g., BIOS, setup program, and one or more option ROMs) are disclosed. The setup program, option ROM, and part of the BIOS are compressed using a lossless compression algorithm. A non-compressible portion of the BIOS includes a decompression algorithm and a shadow RAM

> block table of chipset-specific registers and bit patterns to write-enable and read-enable shadow RAM (RAM that is mapped to the ROM address space). The compressed data is stored in a compressed data block format with the associated location in memory to decompress the compressed data."

Greene, Abstract. *See also* 1:40-1:63.

> "During the BIOS Power-On Self-Test (POST) process, the compressed system ROM is copied to conventional memory, and the decompression program is executed."

Greene, Abstract. *See also* 1:64-2:13.

> "System ROM 210 comprises BIOS (Basic Input/Output System) 310, and optionally setup program 320, and one or more option ROM images for input/output (I/O) devices associated with the computer 332. An "image" is a binary representation of code and/or data of a computer file. BIOS 310 comprises a plurality of routines that initialize the computer hardware and provide primitive I/O services that the operating system and application programs use to manipulate hardware associated with the computer. System setup program 320 is a utility that allows the end user to configure BIOS 310."

Greene, 3:16-28, Fig. 3.

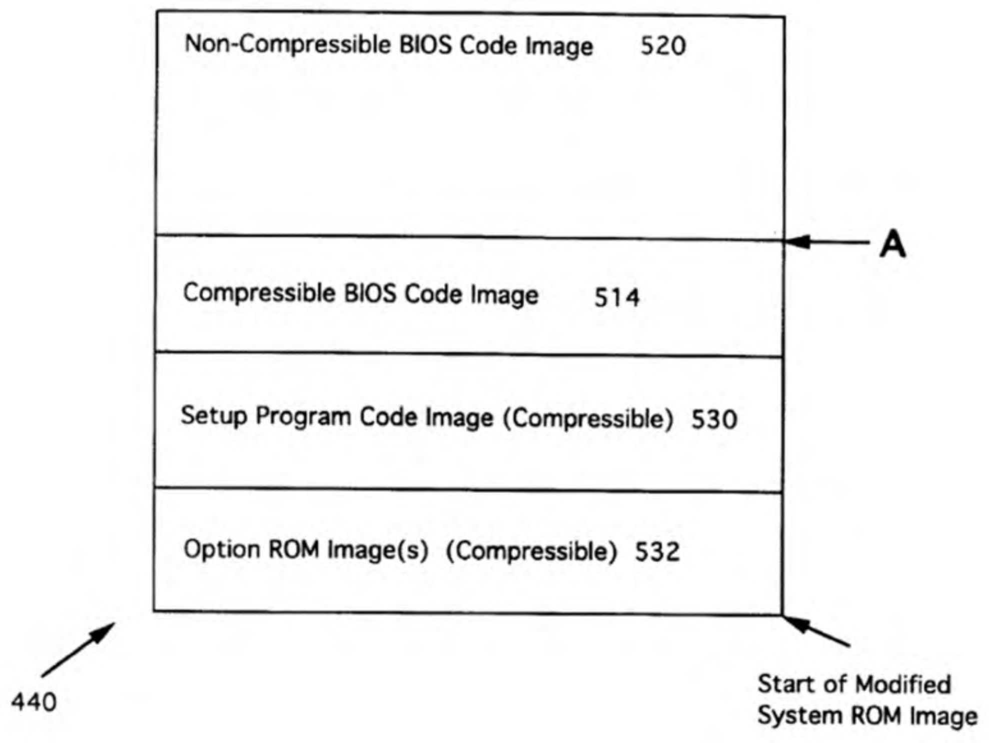
6.1 a memory; and	Greene, as evidenced by the example citations below, discloses "a memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a memory.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 1.1 and 1.2 above.	

6.2 a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor	Greene, as evidenced by the example citations below, discloses "a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Greene discloses this limitation:

See Claims 1.1 and 1.2 above.


Non-Compressible BIOS Code Image 520
A
Compressible BIOS Code Image 514
Setup Program Code Image (Compressible) 530
Option ROM Image(s) (Compressible) 532
440
Start of Modified System ROM Image

Greene, Fig. 5.

> "A chipset (platform)-independent method and apparatus for compressing and decompressing a system ROM of a computer (e.g., BIOS, setup program, and one or more option ROMs) are disclosed.

> The setup program, option ROM, and part of the BIOS are compressed using a lossless compression algorithm. A non-compressible portion of the BIOS includes a decompression algorithm and a shadow RAM block table of chipset-specific registers and bit patterns to write-enable and read-enable shadow RAM (RAM that is mapped to the ROM address space). The compressed data is stored in a compressed data block format with the associated location in memory to decompress the compressed data."

Greene, Abstract. *See also* 1:40-1:63.

> "During the BIOS Power-On Self-Test (POST) process, the compressed system ROM is copied to conventional memory, and the decompression program is executed."

Greene, Abstract. *See also* 1:64-2:13.

> "System ROM 210 comprises BIOS (Basic Input/Output System) 310, and optionally setup program 320, and one or more option ROM images for input/output (I/O) devices associated with the computer 332. An "image" is a binary representation of code and/or data of a computer file. BIOS 310 comprises a plurality of routines that initialize the computer hardware and provide primitive I/O services that the operating system and application programs use to manipulate hardware associated with the computer. System setup program 320 is a utility that allows the end user to configure BIOS 310."

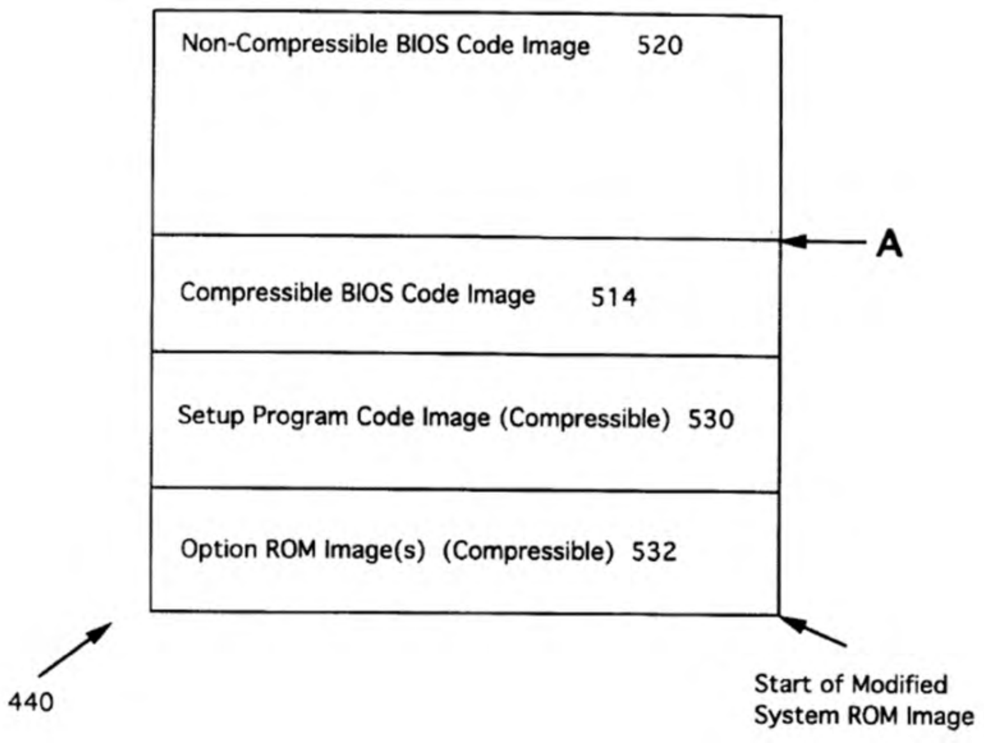
Greene, 3:16-28, Fig. 3.

Appendix C5
Invalidity of U.S. Patent 8,880,862 based on Greene

6.3 wherein the processor is configured:	Greene, as evidenced by the example citations below, discloses "wherein the processor is configured:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the processor is configured), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Greene discloses this limitation:


Non-Compressible BIOS Code Image 520
A
Compressible BIOS Code Image 514
Setup Program Code Image (Compressible) 530
Option ROM Image(s) (Compressible) 532
440
Start of Modified System ROM Image

Greene, Fig. 5.

> "A chipset (platform)-independent method and apparatus for compressing and decompressing a system ROM of a computer (e.g., BIOS, setup program, and one or more option ROMs) are disclosed. The setup program, option ROM, and part of the BIOS are compressed using a lossless compression algorithm. A non-compressible portion of the BIOS includes a decompression algorithm and a shadow RAM block table of chipset-specific registers and bit patterns to write-enable and read-enable shadow RAM (RAM that is mapped to the ROM address space). The compressed data is stored in a compressed data

> block format with the associated location in memory to decompress the compressed data."

Greene, Abstract. *See also* 1:40-1:63.

> "During the BIOS Power-On Self-Test (POST) process, the compressed system ROM is copied to conventional memory, and the decompression program is executed."

Greene, Abstract. *See also* 1:64-2:13.

> "System ROM 210 comprises BIOS (Basic Input/Output System) 310, and optionally setup program 320, and one or more option ROM images for input/output (I/O) devices associated with the computer 332. An "image" is a binary representation of code and/or data of a computer file. BIOS 310 comprises a plurality of routines that initialize the computer hardware and provide primitive I/O services that the operating system and application programs use to manipulate hardware associated with the computer. System setup program 320 is a utility that allows the end user to configure BIOS 310."

Greene, 3:16-28, Fig. 3.

6.4 to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory,	Greene, as evidenced by the example citations below, discloses "to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.1 above.	

6.5 to access the loaded portion of the boot data in the compressed form,	Greene, as evidenced by the example citations below, discloses "to access the loaded portion of the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to access the loaded portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.2 above.	

6.6 to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data, and	Greene, as evidenced by the example citations below, discloses "to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.3 above.	

6.7 to update the boot data list.	Greene, as evidenced by the example citations below, discloses "to update the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to update the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.4.1 above.	

Appendix C5
Invalidity of U.S. Patent 8,880,862 based on Greene

8 (Preamble) A method of loading an operating system for booting a computer system, comprising:	Greene, as evidenced by the example citations below, discloses "A method of loading an operating system for booting a computer system, comprising:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method of loading an operating system for booting a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1 (Preamble) above.	

8.1 storing a portion of the operating system in a compressed form in a first memory;	Greene, as evidenced by the example citations below, discloses "storing a portion of the operating system in a compressed form in a first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing a portion of the operating system in a compressed form in a first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 6.2 above.	

8.2 loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list;	Greene, as evidenced by the example citations below, discloses "loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.1 above.	

8.3 accessing the loaded portion of the operating system from the second memory in the compressed form;	Greene, as evidenced by the example citations below, discloses “accessing the loaded portion of the operating system from the second memory in the compressed form”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded portion of the operating system from the second memory in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.2 above.	

8.4 decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system;	Greene, as evidenced by the example citations below, discloses "decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 1.3 and 1.4.2 above.	

8.5 utilizing the decompressed portion of the operating system to at least partially boot the computer system; and	Greene, as evidenced by the example citations below, discloses "utilizing the decompressed portion of the operating system to at least partially boot the computer system"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed portion of the operating system to at least partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Greene discloses this limitation:

See Claims 1.3 and 1.4.2 above.

See also

> "During the BIOS Power-On Self-Test (POST) process of the target computer, the compressed system ROM image is copied to conventional memory (e.g., RAM), and the decompression program is executed. The decompression program write-enables shadow RAM (with reference to the the chipset-specific information in the shadow RAM block table), copies the non-compressible BIOS code image from conventional memory to shadow RAM, and read-enables shadow RAM. The decompression program scans the compressed system ROM image for compressed data blocks and decompresses the compressed data therein to the associated locations in memory. If data is located in shadow RAM, shadow RAM is write-enabled and read-enabled (with reference to the chipset-specific information in the shadow RAM block table). If compressed data was decompressed to conventional memory, this space is cleared before exiting the POST process."

Greene, 1:64-2:13.

> "Referring to FIG. 10, decompression program 422 comprises a decompression algorithm 1102. Decompression algorithm 1102 corresponds to the compression algorithm used (e.g., LZSS or LZARI)."

Greene, 3:66-4:2.

> "Compressed system ROM 632 image must be decompressed before the compressed data therein can be used on the target computer. In a preferred

> embodiment, decompression is done early in the POST process."

Greene, 7:21-24.

> "Program execution control is transferred 804 to decompression program 422 (in non-compressible BIOS code image 520 of compressed system ROM image 632)."

Greene, 7:29-32, Fig. 8a. *See also* 7:33-7:64.

> "The decompression scan process repeats 832 until each compressed data block 700 in compressed system ROM image 632 is decompressed. In a preferred embodiment, program execution control is then transferred 834 to BIOS 310, now running in shadow RAM 202. In one embodiment, if data was decompressed to conventional (RAM) memory 110, conventional memory 110 is cleared 836 at the end of the POST process and before the operating system is booted (for compatibility reasons)."

Greene, 7:65-8:21.

8.6 updating the boot data list,	Greene, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.4.1 above.	

8.7 wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form.	Greene, as evidenced by the example citations below, discloses "wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 1-1.3 and 5.7 above.	

Appendix C5
Invalidity of U.S. Patent 8,880,862 based on Greene

9.1 The method of claim 8, further comprising: compressing an additional portion of the operating system that is not associated with the boot data list; and	Greene, as evidenced by the example citations below, discloses "the method of claim 8, further comprising: compressing an additional portion of the operating system that is not associated with the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, compressing an additional portion of the operating system that is not associated with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.1 above.	

9.2 storing the additional portion of the operating system in the first memory, and	Greene, as evidenced by the example citations below, discloses "storing the additional portion of the operating system in the first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing the additional portion of the operating system in the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 8.1 above.	

9.3 wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system.	Greene, as evidenced by the example citations below, discloses "wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 8.5 above.	

10. The method of claim 9, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder.	Greene, as evidenced by the example citations below, discloses "the method of claim 9, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Greene discloses this limitation:

See Claim 9.1 above.

> "A lossless compression algorithm is used to compress the modified system ROM image (up to the non-compressible BIOS code image address), thereby generating a compressed system ROM image."

Greene, 1:54-58.

> "Referring now to FIG. 6, modified system ROM image 440 is input to compression utility 600. Compression utility 600 uses compression-related information table 424 (in modified system ROM image 440) to determine 610 a compression algorithm 612 to use in compressing modified system ROM image 440. As discussed above, compression algorithm 612 can be any lossless compression algorithm, but must correspond to the decompression algorithm (1102 below) used in the system ROM."

Greene, 5:19-27, Fig. 6.

Appendix C5
Invalidity of U.S. Patent 8,880,862 based on Greene

11 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	Greene, as evidenced by the example citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, comprising:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1 (Preamble) above.	

11.1 loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system;	Greene, as evidenced by the example citations below, discloses "loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system"

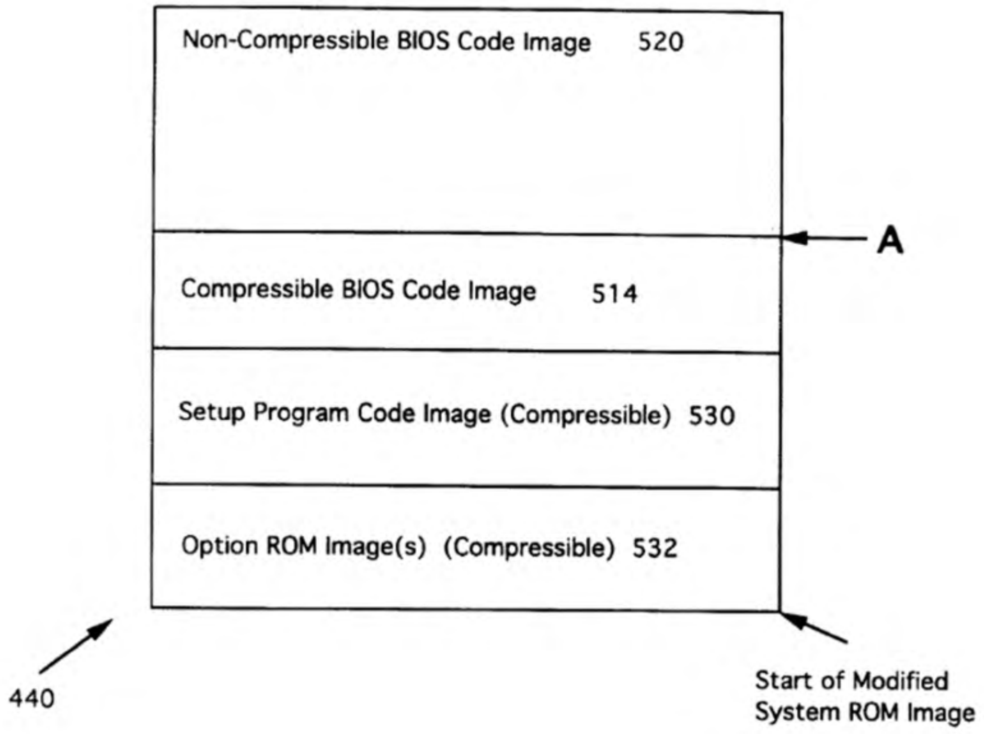
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Greene discloses this limitation:

See Claim 1.1 above.

See also **Look for additional references where loading occurs upon initialization of the computer system**

See also **Add disclosure from '608 Patent Claim 1.2**


Non-Compressible BIOS Code Image 520
A
Compressible BIOS Code Image 514
Setup Program Code Image (Compressible) 530
Option ROM Image(s) (Compressible) 532
440
Start of Modified System ROM Image

Greene, Fig. 5.

> "A chipset (platform)-independent method and apparatus for compressing and decompressing a system ROM of a computer (e.g., BIOS, setup program, and one or more option ROMs) are disclosed. The setup program, option ROM, and part of the BIOS are compressed using a lossless compression algorithm. A non-compressible portion of the BIOS includes a decompression algorithm and a shadow RAM block table of chipset-specific registers and bit patterns to write-enable and read-enable shadow RAM (RAM that is mapped to the ROM address space). The compressed data is stored in a compressed data block format with the associated location in memory to decompress the compressed data."

Greene, Abstract. *See also* 1:40-1:63.

> "During the BIOS Power-On Self-Test (POST) process, the compressed system ROM is copied to conventional memory, and the decompression program is executed."

Greene, Abstract.

> "During the BIOS Power-On Self-Test (POST) process of the target computer, the compressed system ROM image is copied to conventional memory (e.g., RAM), and the decompression program is executed."

Greene, 1:64-67.

> "System ROM 210 comprises BIOS (Basic Input/Output System) 310, and optionally setup program 320, and one or more option ROM images for input/output (I/O) devices associated with the computer 332. An "image" is a binary representation of code and/or data of a computer file. BIOS 310 comprises a plurality of routines that initialize the computer hardware and provide primitive I/O services that the operating system and application programs use to manipulate hardware associated with the computer. System setup program 320 is a utility that allows the end user to configure BIOS 310."

Greene, 3:16-28, Fig. 3.

11.2 accessing the loaded boot data in compressed form from the memory;	Greene, as evidenced by the example citations below, discloses "accessing the loaded boot data in compressed form from the memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded boot data in compressed form from the memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.2 above.	

11.3 decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form;	Greene, as evidenced by the example citations below, discloses "decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.3 above.	

11.4 utilizing the decompressed boot data to load at least a portion of the operating system for the computer system; and	Greene, as evidenced by the example citations below, discloses "utilizing the decompressed boot data to load at least a portion of the operating system for the computer system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed boot data to load at least a portion of the operating system for the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: "During the BIOS Power-On Self-Test (POST) process of the target computer, the compressed system ROM image is copied to conventional memory (e.g., RAM), and the decompression program is executed. The decompression program write-enables shadow RAM (with reference to the the chipset-specific information in the shadow RAM block table), copies the non-compressible BIOS code image from conventional memory to shadow RAM, and read-enables shadow RAM. The decompression program scans the compressed system ROM image for compressed data blocks and decompresses the compressed data therein to the associated locations in memory. If data is located in shadow RAM, shadow RAM is write-enabled and read-enabled (with reference to the chipset-specific information in the shadow RAM block table). If compressed data was decompressed to conventional memory, this space is cleared before exiting the POST process." Greene, 1:64-2:13. "Referring to FIG. 10, decompression program 422 comprises a decompression algorithm 1102. Decompression algorithm 1102 corresponds to the compression algorithm used (e.g., LZSS or LZARI)." Greene, 3:66-4:2. "Compressed system ROM 632 image must be decompressed before the compressed data therein can be used on the target computer. In a preferred embodiment, decompression is done early in the POST process." Greene, 7:21-24. "Program execution control is transferred 804 to decompression program	

> 422 (in non-compressible BIOS code image 520 of compressed system ROM image 632)."

Greene, 7:29-32, Fig. 8a. *See also* 7:33-7:64.

> "The decompression scan process repeats 832 until each compressed data block 700 in compressed system ROM image 632 is decompressed. In a preferred embodiment, program execution control is then transferred 834 to BIOS 310, now running in shadow RAM 202. In one embodiment, if data was decompressed to conventional (RAM) memory 110, conventional memory 110 is cleared 836 at the end of the POST process and before the operating system is booted (for compatibility reasons)."

Greene, 7:65-8:21.

11.5 updating the boot data list.	Greene, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.4.1 above.	

Appendix C5
Invalidity of U.S. Patent 8,880,862 based on Greene

13 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	Greene, as evidenced by the example citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, comprising"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1 (Preamble) above.	

13.1 loading boot data in a compressed form that is associated with a boot data list from a boot device;	Greene, as evidenced by the example citations below, discloses "loading boot data in a compressed form that is associated with a boot data list from a boot device"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading boot data in a compressed form that is associated with a boot data list from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.1 above.	

13.2 accessing the loaded boot data in the compressed form;	Greene, as evidenced by the example citations below, discloses "accessing the loaded boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.2 above.	

13.3 decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form;	Greene, as evidenced by the example citations below, discloses "decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.3 above.	

13.4 updating the boot data list.	Greene, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.4.1 above.	

Appendix C5
Invalidity of U.S. Patent 8,880,862 based on Greene

14 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	Greene, as evidenced by the example citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, comprising"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1 (Preamble) above.	

Appendix C5
Invalidity of U.S. Patent 8,880,862 based on Greene

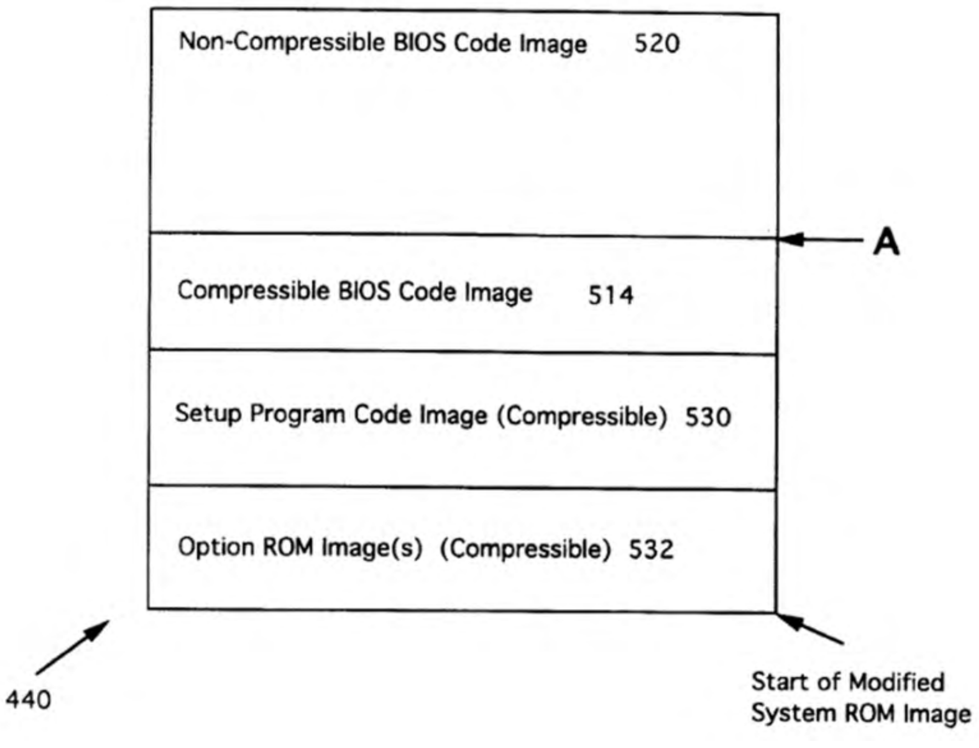
14.1 accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list;	Greene, as evidenced by the example citations below, discloses “accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list”

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.

Greene discloses this limitation:

See Claims 1.1 and 1.2 above.

See ***Add disclosure from ‘608 Patent Claim 1.1***


Non-Compressible BIOS Code Image 520
A
Compressible BIOS Code Image 514
Setup Program Code Image (Compressible) 530
Option ROM Image(s) (Compressible) 532
440
Start of Modified System ROM Image

Greene, Fig. 5.

“A chipset (platform)-independent method and apparatus for

> compressing and decompressing a system ROM of a computer (e.g., BIOS, setup program, and one or more option ROMs) are disclosed. The setup program, option ROM, and part of the BIOS are compressed using a lossless compression algorithm. A non-compressible portion of the BIOS includes a decompression algorithm and a shadow RAM block table of chipset-specific registers and bit patterns to write-enable and read-enable shadow RAM (RAM that is mapped to the ROM address space). The compressed data is stored in a compressed data block format with the associated location in memory to decompress the compressed data."

Greene, Abstract. *See also* 1:40-1:63.

> "During the BIOS Power-On Self-Test (POST) process, the compressed system ROM is copied to conventional memory, and the decompression program is executed."

Greene, Abstract. *See also* 1:64-2:13.

> "System ROM 210 comprises BIOS (Basic Input/Output System) 310, and optionally setup program 320, and one or more option ROM images for input/output (I/O) devices associated with the computer 332. An "image" is a binary representation of code and/or data of a computer file. BIOS 310 comprises a plurality of routines that initialize the computer hardware and provide primitive I/O services that the operating system and application programs use to manipulate hardware associated with the computer. System setup program 320 is a utility that allows the end user to configure BIOS 310."

Greene, 3:16-28, Fig. 3.

> "BIOS code 410, and optionally setup program code 430, and one or more option ROM code modules 432, are each developed, written (e.g., in any computer language such as C or Assembly language) and saved in a computer file. Code and data, as referred to herein, are used interchangeably and comprise computer code and/or data. BIOS code 410 is separated 412 into compressible BIOS code 414 and non-compressible BIOS code 420. Non-compressible BIOS code 420 comprises compression-related information table 424, decompression program 422, and shadow RAM block table 1104."

Greene, 3:42-52.

> "Continuing with FIG. 4, compressible BIOS code 414, non-compressible BIOS code 420 (comprising decompression program 422

(1102, 1106, 1108), compression-related information table 424, and shadow RAM block table 1104), and optionally setup program code 430, and option ROM code module(s) 432, are compiled or assembled and linked 434 to generate a modified system ROM image 440." Greene, 4:64-5:3.

14.2 loading the boot data into a memory; and	Greene, as evidenced by the example citations below, discloses "loading the boot data into a memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the boot data into a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.1 above.	

14.3 servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form; and	Greene, as evidenced by the example citations below, discloses "servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Greene discloses this limitation:

See Claims 1.2 and 1.3 above.

> "The compressed data is stored in a compressed data block format with the associated location in memory to decompress the compressed data. Thus, the data can be decompressed anywhere in memory of a target computer. For example, the BIOS is decompressed to shadow RAM and the setup program is decompressed to conventional memory."

Greene, Abstract.

> "The decompression program scans the compressed system ROM for compressed data blocks and decompresses the compressed data therein to the associated locations in memory. If compressed data is located in shadow RAM, shadow RAM is enabled for writing and reading with reference to the chipset-specific information in the shadow RAM block table. If compressed data was decompressed to conventional memory space, this space is cleared before exiting the POST process."

Greene, Abstract.

> "During the BIOS Power-On Self-Test (POST) process of the target computer, the compressed system ROM image is copied to conventional

memory (e.g., RAM), and the decompression program is executed. The decompression program write-enables shadow RAM (with reference to the the chipset-specific information in the shadow RAM block table), copies the non-compressible BIOS code image from conventional memory to shadow RAM, and read-enables shadow RAM. The decompression program scans the compressed system ROM image for compressed data blocks and decompresses the compressed data therein to the associated locations in memory. If data is located in shadow RAM, shadow RAM is write-enabled and read-enabled (with reference to the chipset-specific information in the shadow RAM block table). If compressed data was decompressed to conventional memory, this space is cleared before exiting the POST process."

Greene, 1:64-2:13.

> "Referring to FIG. 10, decompression program 422 comprises a decompression algorithm 1102. Decompression algorithm 1102 corresponds to the compression algorithm used (e.g., LZSS or LZARI)."

Greene, 3:66-4:2.

> "Compressed system ROM 632 image must be decompressed before the compressed data therein can be used on the target computer. In a preferred embodiment, decompression is done early in the POST process."

Greene, 7:21-24.

> "Program execution control is transferred 804 to decompression program 422 (in non-compressible BIOS code image 520 of compressed system ROM image 632)."

Greene, 7:29-32, Fig. 8a. *See also* 7:33-7:64.

> "The decompression scan process repeats 832 until each compressed data block 700 in compressed system ROM image 632 is decompressed. In a preferred embodiment, program execution control is then transferred 834 to BIOS 310, now running in shadow RAM 202. In one embodiment, if data was decompressed to conventional (RAM) memory 110, conventional memory 110 is cleared 836 at the end of the POST process and before the operating system is booted (for compatibility reasons)."

Greene, 7:65-8:21.

14.4 updating the boot data list.	Greene, as evidenced by the example citations below, discloses “updating the boot data list”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Greene discloses this limitation: *See* Claim 1.4.1 above.	

15. The method of claim 14, wherein the boot data comprises: a program code associated with the operating system.	Greene, as evidenced by the example citations below, discloses "the method of claim 14, wherein the boot data comprises: a program code associated with the operating system."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the boot data comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Greene discloses this limitation:

See Claims 1 (Preamble) and 1.1 above.

16. The method of claim 14, wherein the operating system comprises: a plurality of files.	Greene, as evidenced by the example citations below, discloses "the method of claim 14, wherein the operating system comprises: a plurality of files."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: "Conventional memory 110 comprises a random access memory (RAM) 112 address space that is set aside for use by operating systems and application programs." Greene, 2:50-53. "BIOS 310 comprises a plurality of routines that initialize the computer hardware and provide primitive I/O services that the operating system and application programs use to manipulate hardware associated with the computer." Greene, 3:21-25. "In one embodiment, if data Was decompressed to conventional memory 110, conventional memory 110 is cleared 836 at the end of the POST process and before the operating system is booted (for compatibility reasons)." Greene, 8:20-21.	

17. The method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program.	Greene, as evidenced by the example citations below, discloses "the method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 15 above.	

19.1 The method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:	Greene, as evidenced by the example citations below, discloses "the method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

19.2 associating the accessed boot data that is not associated with the boot data list to the boot data list.	Greene, as evidenced by the example citations below, discloses "associating the accessed boot data that is not associated with the boot data list to the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, associating the accessed boot data that is not associated with the boot data list to the boot data list.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 2 above.	

23. The method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files.	Greene, as evidenced by the example citations below, discloses "the method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 16 above.	

24. The method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system.	Greene, as evidenced by the example citations below, discloses “the method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system.
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Greene discloses this limitation: *See* Claim 15 above.	

27. The method of claim 1, wherein the memory comprises: a physical memory.	Greene, as evidenced by the example citations below, discloses "the method of claim 1, wherein the memory comprises: a physical memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 1.1 and 1.2 above.	

28. The method of claim 1, wherein the operating system comprises: a plurality of files.	Greene, as evidenced by the example citations below, discloses "the method of claim 1, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 16 above.	

29. The method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program.	Greene, as evidenced by the example citations below, discloses "the method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 15 and 17 above.	

31. The method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access.	Greene, as evidenced by the example citations below, discloses "the method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 1.1 and 1.2 above.	

32. The method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form.	Greene, as evidenced by the example citations below, discloses "the method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Greene discloses this limitation:

> "A lossless compression algorithm is used to compress the modified system ROM image (up to the non-compressible BIOS code image address), thereby generating a compressed system ROM image."

Greene, 1:54-58.

> "In a preferred embodiment, compression algorithm is a lossless decompression algorithm, for example, LZSS or LZARI, which are commercially available compression/ decompression algorithms. In general, LZSS decompresses faster than LZARI. LZARI generally compresses data better (smaller) than LZSS. For performance reasons, it is suggested that LZARI decompression code be used with an Intel 80486 or better CPU."

Greene, 3:56-63.

> "Referring to FIG. 10, decompression program 422 comprises a decompression algorithm 1102. Decompression algorithm 1102 corresponds to the compression algorithm used (e.g., LZSS or LZARI)."

Greene, 3:66-4:2.

> "As discussed above, compression algorithm 612 can be any lossless compression algorithm, but must correspond to the decompression algorithm (1102 below) used in the system ROM."

Greene, 5:24-27.

33. The method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form.	Greene, as evidenced by the example citations below, discloses "the method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Greene discloses this limitation:

See Claim 32 above.

> "A lossless compression algorithm is used to compress the modified system ROM image (up to the non-compressible BIOS code image address), thereby generating a compressed system ROM image."

Greene, 1:54-58.

> "In a preferred embodiment, compression algorithm is a lossless decompression algorithm, for example, LZSS or LZARI, which are commercially available compression/ decompression algorithms. In general, LZSS decompresses faster than LZARI. LZARI generally compresses data better (smaller) than LZSS. For performance reasons, it is suggested that LZARI decompression code be used with an Intel 80486 or better CPU."

Greene, 3:56-63.

> "Referring to FIG. 10, decompression program 422 comprises a decompression algorithm 1102. Decompression algorithm 1102 corresponds to the compression algorithm used (e.g., LZSS or LZARI)."

Greene, 3:66-4:2.

> "As discussed above, compression algorithm 612 can be any lossless compression algorithm, but must correspond to the decompression algorithm (1102 below) used in the system ROM."

Greene, 5:24-27.

35. The method of claim 5, wherein the compressed boot data represents a plurality of files.	Greene, as evidenced by the example citations below, discloses "the method of claim 5, wherein the compressed boot data represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the compressed boot data represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 16 and 23 above.	

36. The method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system.	Greene, as evidenced by the example citations below, discloses "the method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 15 and 17 above.	

39. The method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory.	Greene, as evidenced by the example citations below, discloses "the method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 1.1 and 1.2 above.	

40. The method of claim 36, wherein the operating system comprises: a plurality of files.	Greene, as evidenced by the example citations below, discloses "the method of claim 36, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 36, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 16 and 23 above.	

43. The method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access.	Greene, as evidenced by the example citations below, discloses "the method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 31 above.	

44. The method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data.	Greene, as evidenced by the example citations below, discloses "the method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Greene discloses this limitation:

See Claim 32 above.

45. The method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data.	Greene, as evidenced by the example citations below, discloses "the method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 32 and 33 above.	

47. The system of claim 6, wherein the boot data in the compressed form represents a plurality of files.	Greene, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 16 and 23 above.	

48. The system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system.	Greene, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 15, 17 and 24 above.	

49. The system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form.	Greene, as evidenced by the example citations below, discloses "the system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 10 above.	

50. The system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form.	Greene, as evidenced by the example citations below, discloses “the system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Greene discloses this limitation: “Compressed system ROM 632 image must be decompressed before the compressed data therein can be used on the target computer. In a preferred embodiment, decompression is done early in the POST process.” Greene, 7:21-25. See also Greene, 7:26-8:22.	

51. The system of claim 6, wherein the first memory comprises: a physical memory.	Greene, as evidenced by the example citations below, discloses "the system of claim 6, wherein the first memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the first memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 27 and 39 above.	

52. The system of claim 6, wherein the boot data the compressed form comprises: a plurality of files.	Greene, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data the compressed form comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data the compressed form comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 16 and 23 above.	

53. The system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program.	Greene, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 15, 17 and 24 above.	

59. The method of claim 8, wherein the operating system in the compressed form represents a plurality of files.	Greene, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 16 and 23 above.	

60. The method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system.	Greene, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 15, 17 and 24 above.	

63. The method of claim 8, wherein the second memory comprises: a physical memory.	Greene, as evidenced by the example citations below, discloses "the method of claim 8, wherein the second memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the second memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 27 and 38 above.	

64. The method of claim 8, wherein the operating system comprises: a plurality of files.	Greene, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 16 and 23 above.	

65. The method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program.	Greene, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 15, 17 and 24 above.	

67. The method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access.	Greene, as evidenced by the example citations below, discloses "the method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 31 above.	

71. The method of claim 11, wherein the boot data in the compressed form represents a plurality of files.	Greene, as evidenced by the example citations below, discloses "the method of claim 11, wherein the boot data in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 16 and 23 above.	

72. The method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system.	Greene, as evidenced by the example citations below, discloses "the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 15, 17 and 24 above.	

75. The method of claim 11, wherein the memory comprises: a physical memory.	Greene, as evidenced by the example citations below, discloses "the method of claim 11, wherein the memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 27 above.	

76. The method of claim 11, wherein the operating system comprises: a plurality of files.	Greene, as evidenced by the example citations below, discloses "the method of claim 11, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 16 and 23 above.	

77. The method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program.	Greene, as evidenced by the example citations below, discloses "the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 15, 17 and 24 above.	

79. The method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access.	Greene, as evidenced by the example citations below, discloses "the method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 31 above.	

80. The method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form.	Greene, as evidenced by the example citations below, discloses "the method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 32, 33 and 49 above.	

81. The method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form.	Greene, as evidenced by the example citations below, discloses "the method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 32, 33 and 49 above.	

83. The method of claim 13, wherein the boot data in the compressed form represents a plurality of files.	Greene, as evidenced by the example citations below, discloses "the method of claim 13, wherein the boot data in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 16 and 23 above.	

84. The method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system.	Greene, as evidenced by the example citations below, discloses "the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 15, 17 and 24 above.	

87. The method of claim 13, wherein the memory comprises: a physical memory.	Greene, as evidenced by the example citations below, discloses "the method of claim 13, wherein the memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 27 above.	

88. The method of claim 13, wherein the operating system comprises: a plurality of files.	Greene, as evidenced by the example citations below, discloses "the method of claim 13, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 16 and 23 above.	

89. The method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program.	Greene, as evidenced by the example citations below, discloses "the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 15, 17 and 24 above.	

91. The method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access.	Greene, as evidenced by the example citations below, discloses "the method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 31 above.	

92. The method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data.	Greene, as evidenced by the example citations below, discloses "the method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 32, 33 and 49 above.	

93. The method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data.	Greene, as evidenced by the example citations below, discloses "the method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claim 32, 33, and 49 above.	

97.1 The method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list,	Greene, as evidenced by the example citations below, discloses "the method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 9.1-9.3 and 19 above.	

97.2 and wherein the updating comprises: associating the additional compressed boot data with the boot data list.	Greene, as evidenced by the example citations below, discloses "and wherein the updating comprises: associating the additional compressed boot data with the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, and wherein the updating comprises: associating the additional compressed boot data with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 1.4.1, 2, 4, 5.6, and 8.6 above.	

98. The method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list.	Greene, as evidenced by the example citations below, discloses "the method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 1.4.1, 5.6, and 8.6 above.	

107. The method of claim 2, further comprising: storing the updated boot list in a non-volatile memory.	Greene, as evidenced by the example citations below, discloses "the method of claim 2, further comprising: storing the updated boot list in a non-volatile memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 2, further comprising: storing the updated boot list in a non-volatile memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 1.4.1, 5.6, and 8.6 above.	

108. The method of claim 2, further comprising: compressing at least a portion of the additional boot data.	Greene, as evidenced by the example citations below, discloses "the method of claim 2, further comprising: compressing at least a portion of the additional boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 2, further comprising: compressing at least a portion of the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 1.1, 5, 9.1 and 8 above.	

109. The method of claim 108, further comprising: storing the compressed additional boot data.	Greene, as evidenced by the example citations below, discloses "the method of claim 108, further comprising: storing the compressed additional boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 108, further comprising: storing the compressed additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Greene discloses this limitation: *See* Claims 5.1, 8.1 and 9.1 above.	

Appendix C6
Invalidity of U.S. Patent 8,880,862 based on Hillis

U.S. Patent No. 6,421,776 to Hillis ("Hillis") invalidates claims 1-6, 8-11, 13-17, 19, 23-24, 27-29, 31-33, 35-36, 39-40, 43-45, 47-53, 59-60, 63-65, 67, 71-72, 75-77, 79-81, 83-84, 87-89, 91-93, 97-98, and 107-109 of United States Patent No. 8,880,862 ("the '862 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '862 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix C6
Invalidity of U.S. Patent 8,880,862 based on Hillis

1 (Preamble) A method for providing accelerated loading of an operating system in a computer system, the method comprising:	Hillis, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, the method comprising:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system in a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: "Another advantage is in reducing the time required for bootstrapping." Hillis, 3:20-21. "As has been described, this invention enables compression of most of the contents of the BIOS ROM and boot strapping of the system upon BIOS code decompression, by storing only the initial portion of the POST code in BIOS ROM, sufficient to enable the system memory." Hillis, 7:66-8:3.	

1.1 loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory;	Hillis, as evidenced by the example citations below, discloses "loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hillis discloses this limitation:

> "To increase the effective capacity of BIOS, an initial portion of the power on system reset (POST) code that is required to enable the system memory is stored in ROM in uncompressed form, and substantially the remaining portion of the BIOS code is stored in compressed form."

Hillis, Abstract.

> "Upon system initialization during a cold boot, the uncompressed portion of POST is executed from the ROM to enable the system memory, and then an image of the BIOS code is written to shadow memory."

Hillis, Abstract.

> "After locating a disk with a valid boot record, the BIOS program reads the data stored on the first sector of the disk, and copies that data to specific locations in RAM. This information, found in the same location on every formatted disk, constitutes the DOS boot record. The BIOS then passes control to the boot record which instructs the PC on how to load the two hidden operating system files to RAM (the files named IBMBIO.COM and IBMDOS.COM on IBM computers). After loading other operating system files into RAM to carry out the rest of the boot up sequence, the boot record is no longer needed."

Hillis, 1:46-56.

> "To reduce the time required for decompression of BIOS code, the code is transferred from ROM to the system memory in compressed form."

Hillis, 3:49-51.

> "Upon cold boot, the initial portion POST is read directly from ROM to enable the system memory, and then an image of the entire BIOS code, the major portion of which is in compressed form, is written to RAM in the system memory, and control is transferred to the image."

Hillis, 3:54-58.

> "The next layers of the BIOS ROM consist of compressed set-up data, including descriptive text, and compressed setup code. Next, all but the initial portion of the power on system test (POST) code (termed "phase 2 POST herein") is stored in compressed form, the initial portion of POST (termed "phase 1 POST") being stored in the next lower layer of BIOS ROM."

Hillis, 5:35-41.

> "Then, all the remaining portion of POST and other BIOS code are copied to memory in a region thereof termed "shadow RAM" or "shadow memory."

Hillis, 5:48-55.

> "As shall be described in more detail hereinafter, upon system initialization (bootstrapping), an image of the BIOS ROM is copied to the upper 128 kbyte region of system memory, or shadow RAM, from where system execution takes place for higher operating speed. Shadowing of the BIOS is well known. In accordance with the invention, however, most of the BIOS code is stored in ROM in compressed form."

Hillis, 6:1-8.

> "Next, for improved performance the entire ROM image is transferred to the system memory (step 50) and the microprocessor cache is turned on (step 52). Control of the system is now transferred to the RAM image of POST, shown in FIG. 4 (step 54)."

Hillis, 6:50-54.

> "Next, the uncompressed images are copied from the current region of system RAM back into the shadow RAM (step 58)"

Hillis, 6:61-63.

> "As has been described, this invention enables compression of most of the contents of the BIOS ROM and boot strapping of the system upon BIOS code decompression, by storing only the initial portion of the POST code in BIOS ROM, sufficient to enable the system memory."

Hillis, 7:66-8:3.

1.2 accessing the loaded portion of the boot data in the compressed form from memory;	Hillis, as evidenced by the example citations below, discloses "accessing the loaded portion of the boot data in the compressed form from memory;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded portion of the boot data in the compressed form from memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hillis discloses this limitation:

> "Then, after a jump from one location of the system memory to another, decompression of the code takes place."

Hillis, 3:51-53.

> "As needed, portions of the BIOS code including POST, Setup (if invoked) and then other BIOS routines are selectively decompressed from the shadow memory to another location of the system memory. Normal execution of POST and BIOS then proceeds until the boot is completed."

Hillis, 3:58-63.

> "Mapping from the BIOS ROM to the system memory is followed by decompression only of those BIOS routines that are necessary. Referring to FIG. 5 depicting the BIOS decompressed shadow RAM image, the upper 64K of shadow memory contains decompressed BIOS code, the lowest 32K portion of the upper 128 kbyte block contains decompressed phase 2 POST code, decompressed set-up data and code reside at the lowest 128K in system memory, and decompressed video and SES reside as shown."

Hillis, 6:8-16.

> "As BIOS routines are needed, they are now selectively decompressed from system RAM to system RAM (step 56). The POST may be decompressed into any other region of RAM within or outside the system RAM range of addresses including the region ultimately to be occupied by the operating system."

Hillis, 6:55-60. "As has been described, this invention enables compression of most of the contents of the BIOS ROM and boot strapping of the system upon BIOS code decompression, by storing only the initial portion of the POST code in BIOS ROM, sufficient to enable the system memory." Hillis, 7:66-8:3.

1.3 decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form; and	Hillis, as evidenced by the example citations below, discloses "decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form; and"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: "As BIOS code is needed during the remainder of the boot, the code is selectively decompressed from the shadow memory to another region of the system memory to which control is transferred." Hillis, Abstract. "A further advantage is in performing BIOS code decompression under different boot sceneries, cold and warm, and upon memory conditions of real and protect." Hillis, 3:21-24. "Then, after a jump from one location of the system memory to another, decompression of the code takes place." Hillis, 3:51-53. "As needed, portions of the BIOS code including POST, Setup (if invoked) and then other BIOS routines are selectively decompressed from the shadow memory to another location of the system memory. Normal execution of POST and BIOS then proceeds until the boot is completed." Hillis, 3:58-63. "Mapping from the BIOS ROM to the system memory is followed by	

> decompression only of those BIOS routines that are necessary. Referring to FIG. 5 depicting the BIOS decompressed shadow RAM image, the upper 64K of shadow memory contains decompressed BIOS code, the lowest 32K portion of the upper 128 kbyte block contains decompressed phase 2 POST code, decompressed set-up data and code reside at the lowest 128K in system memory, and decompressed video and SES reside as shown."

Hillis, 6:8-16.

> "As BIOS routines are needed, they are now selectively decompressed from system RAM to system RAM (step 56). The POST may be decompressed into any other region of RAM within or outside the system RAM range of addresses including the region ultimately to be occupied by the operating system."

Hillis, 6:55-60.

> "Decompression occurs selectively in a similar manner for other compressed images of the BIOS code, other than setup which in this example has not yet been called (step 60). Control of the system is transferred to the shadow image of POST (step 62)."

Hillis, 6:63-67.

> "As has been described, this invention enables compression of most of the contents of the BIOS ROM and boot strapping of the system upon BIOS code decompression, by storing only the initial portion of the POST code in BIOS ROM, sufficient to enable the system memory."

Hillis, 7:66-8:3.

1.4.1 updating the boot data list,	Hillis, as evidenced by the example citations below, discloses “updating the boot data list,”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.	

1.4.2 wherein the decompressed portion of boot data comprises a portion of the operating system.	Hillis, as evidenced by the example citations below, discloses "wherein the decompressed portion of boot data comprises a portion of the operating system."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the decompressed portion of boot data comprises a portion of the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hillis discloses this limitation:

> "A further advantage is in performing BIOS code decompression under different boot sceneries, cold and warm, and upon memory conditions of real and protect."

Hillis, 3:21-24.

> "As needed, portions of the BIOS code including POST, Setup (if invoked) and then other BIOS routines are selectively decompressed from the shadow memory to another location of the system memory. Normal execution of POST and BIOS then proceeds until the boot is completed."

Hillis, 3:58-63.

> "As has been described, this invention enables compression of most of the contents of the BIOS ROM and boot strapping of the system upon BIOS code decompression, by storing only the initial portion of the POST code in BIOS ROM, sufficient to enable the system memory."

Hillis, 7:66-8:3.

2. The method of claim 1, wherein the updating comprises: associating additional boot data with the boot data list.	Hillis, as evidenced by the example citations below, discloses "wherein the updating comprises: associating additional boot data with the boot data list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein associating additional boot data with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1.4.1 above.	

3. The method of claim 1, wherein the updating comprises: removing an association of additional boot data that is associated with the boot data list from the boot data list.	Hillis, as evidenced by the example citations below, discloses "wherein the updating comprises: removing an association of additional boot data that is associated with the boot data list from the boot data list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein removing an association of additional boot data that is associated with the boot data list from the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1.4.1 above.	

4. The method of claim 1, wherein the updating comprises: associating additional boot data with the boot data list; and compressing a portion of the additional boot data.	Hillis, as evidenced by the example citations below, discloses "wherein the updating comprises: associating additional boot data with the boot data list; and compressing a portion of the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein associating additional boot data with the boot data list; and compressing a portion of the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 1.4.1, and 2 above.	

5 (Preamble) A method for booting a computer system, the method comprising:	Hillis, as evidenced by the example citations below, discloses "a method for booting a computer system, the method comprising:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hillis discloses this limitation:

See Claims 1-1.4.2 above.

See also ***Add disclosure from '608 Patent Claims 1.1 and 1.2.***

> "After locating a disk with a valid boot record, the BIOS program reads the data stored on the first sector of the disk, and copies that data to specific locations in RAM. This information, found in the same location on every formatted disk, constitutes the DOS boot record. The BIOS then passes control to the boot record which instructs the PC on how to load the two hidden operating system files to RAM (the files named IBMBIO.COM and IBMDOS.COM on IBM computers). After loading other operating system files into RAM to carry out the rest of the boot up sequence, the boot record is no longer needed."

Hillis, 1:46-56.

> "Upon system initialization during a cold boot, the uncompressed portion of POST is executed from the ROM to enable the system memory, and then an image of the BIOS code is written to shadow memory."

Hillis, Abstract.

> "In accordance with an aspect of the invention, the portion of the BIOS code that is uncompressed in ROM includes an initial portion of a power on system test (POST) code which is sufficient to enable the system memory, a remaining portion of which is compressed."

Hillis, 3:44-48.

> "Upon cold boot, the initial portion POST is read directly from ROM to enable the system memory, and then an image of the entire BIOS code, the major portion of which is in compressed form, is written to RAM in

> the system memory, and control is transferred to the image."

Hillis, 3:54-58.

> "The phase 1 POST code, which is stored in uncompressed (unpacked) form, consists of only that portion of POST that is necessary to enable the system memory. That is, under conventional BIOS protocol, the initial portion of POST is first read from the BIOS ROM to enable or "wake up" the system memory, usually composed of CMOS type random access semiconductor memory."

Hillis, 3:42-48.

> "As shall be described in more detail hereinafter, upon system initialization (bootstrapping), an image of the BIOS ROM is copied to the upper 128 kbyte region of system memory, or shadow RAM, from Where system execution takes place for higher operating speed."

Hillis: 6:1-4.

See also Hillis, 6:16-49.

5.1 storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory;	Hillis, as evidenced by the example citations below, discloses "storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1.1 above. *See also* "To increase the effective capacity of BIOS, an initial portion of the power on system reset (POST) code that is required to enable the system memory is stored in ROM in uncompressed form, and substantially the remaining portion of the BIOS code is stored in compressed form." Hillis, Abstract.	

5.2 loading the stored compressed boot data from the first memory;	Hillis, as evidenced by the example citations below, discloses "loading the stored compressed boot data from the first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the stored compressed boot data from the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1.1 above.	

5.3 accessing the loaded compressed boot data;	Hillis, as evidenced by the example citations below, discloses "accessing the loaded compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1.2 above.	

5.4 decompressing the accessed compressed boot data;	Hillis, as evidenced by the example citations below, discloses "decompressing the accessed compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1.3 above.	

5.5 utilizing the decompressed boot data to at least partially boot the computer system;	Hillis, as evidenced by the example citations below, discloses "utilizing the decompressed boot data to at least partially boot the computer system"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed boot data to at least partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hillis discloses this limitation:

See Claims 1.3 and 1.4.2 above.

See also

> "A further advantage is in performing BIOS code decompression under different boot sceneries, cold and warm, and upon memory conditions of real and protect."

Hillis, 3:21-24.

> "As needed, portions of the BIOS code including POST, Setup (if invoked) and then other BIOS routines are selectively decompressed from the shadow memory to another location of the system memory. Normal execution of POST and BIOS then proceeds until the boot is completed."

Hillis, 3:58-63.

> "As BIOS routines are needed, they are now selectively decompressed from system RAM to system RAM (step 56). The POST may be decompressed into any other region of RAM within or outside the system RAM range of addresses including the region ultimately to be occupied by the operating system."

Hillis, 6:55-60.

> "Decompression occurs selectively in a similar manner for other compressed images of the BIOS code, other than setup which in this example has not yet been called (step 60). Control of the system is transferred to the shadow image of POST (step 62)."

Hillis, 6:63-67. "As has been described, this invention enables compression of most of the contents of the BIOS ROM and boot strapping of the system upon BIOS code decompression, by storing only the initial portion of the POST code in BIOS ROM, sufficient to enable the system memory." Hillis, 7:66-8:3.

5.6 updating the boot data list;	Hillis, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1.4.1 above.	

5.7 wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form.	Hillis, as evidenced by the example citations below, discloses "wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 1-1.3 above.	

6 (Preamble) A system comprising: a processor;	Hillis, as evidenced by the example citations below, discloses "a system comprising: a processor:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system comprising: a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hillis discloses this limitation:

> "Upon system initialization during a cold boot, the uncompressed portion of POST is executed from the ROM to enable the system memory, and then an image of the BIOS code is written to shadow memory."

Hillis, Abstract.

> "In accordance with an aspect of the invention, the portion of the BIOS code that is uncompressed in ROM includes an initial portion of a power on system test (POST) code which is sufficient to enable the system memory, a remaining portion of which is compressed."

Hillis, 3:44-48.

> "Upon cold boot, the initial portion POST is read directly from ROM to enable the system memory, and then an image of the entire BIOS code, the major portion of which is in compressed form, is written to RAM in the system memory, and control is transferred to the image."

Hillis, 3:54-58.

> "The phase 1 POST code, which is stored in uncompressed (unpacked) form, consists of only that portion of POST that is necessary to enable the system memory. That is, under conventional BIOS protocol, the initial portion of POST is first read from the BIOS ROM to enable or "wake up" the system memory, usually composed of CMOS type random access semiconductor memory."

Hillis, 3:42-48.

> "As shall be described in more detail hereinafter, upon system initialization (bootstrapping), an image of the BIOS ROM is copied to the upper 128 kbyte region of system memory, or shadow RAM, from Where

system execution takes place for higher operating speed." Hillis: 6:1-4. *See also* Hillis, 6:16-49.

6.1 a memory; and	Hillis, as evidenced by the example citations below, discloses “a memory”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a memory.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Hillis discloses this limitation: *See* Claims 1.1 and 1.2 above.	

6.2 a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor	Hillis, as evidenced by the example citations below, discloses "a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hillis discloses this limitation:

See Claims 1.1 and 1.2 above.

See also

> "Upon system initialization during a cold boot, the uncompressed portion of POST is executed from the ROM to enable the system memory, and then an image of the BIOS code is written to shadow memory."

Hillis, Abstract.

> "In accordance with an aspect of the invention, the portion of the BIOS code that is uncompressed in ROM includes an initial portion of a power on system test (POST) code which is sufficient to enable the system memory, a remaining portion of which is compressed."

Hillis, 3:44-48.

> "Upon cold boot, the initial portion POST is read directly from ROM to enable the system memory, and then an image of the entire BIOS code, the major portion of which is in compressed form, is written to RAM in the system memory, and control is transferred to the image."

Hillis, 3:54-58.

> "The phase 1 POST code, which is stored in uncompressed (unpacked) form, consists of only that portion of POST that is necessary to enable the system memory. That is, under conventional BIOS protocol, the initial portion of POST is first read from the BIOS ROM to enable or "wake up" the system memory, usually composed of CMOS type random access

semiconductor memory."

Hillis, 3:42-48.

> "As shall be described in more detail hereinafter, upon system initialization (bootstrapping), an image of the BIOS ROM is copied to the upper 128 kbyte region of system memory, or shadow RAM, from Where system execution takes place for higher operating speed."

Hillis: 6:1-4.

See also Hillis, 6:16-49.

6.3 wherein the processor is configured:	Hillis, as evidenced by the example citations below, discloses "wherein the processor is configured:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the processor is configured), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hillis discloses this limitation:

> "Upon system initialization during a cold boot, the uncompressed portion of POST is executed from the ROM to enable the system memory, and then an image of the BIOS code is written to shadow memory."

Hillis, Abstract.

> "In accordance with an aspect of the invention, the portion of the BIOS code that is uncompressed in ROM includes an initial portion of a power on system test (POST) code which is sufficient to enable the system memory, a remaining portion of which is compressed."

Hillis, 3:44-48.

> "Upon cold boot, the initial portion POST is read directly from ROM to enable the system memory, and then an image of the entire BIOS code, the major portion of which is in compressed form, is written to RAM in the system memory, and control is transferred to the image."

Hillis, 3:54-58.

> "The phase 1 POST code, which is stored in uncompressed (unpacked) form, consists of only that portion of POST that is necessary to enable the system memory. That is, under conventional BIOS protocol, the initial portion of POST is first read from the BIOS ROM to enable or "wake up" the system memory, usually composed of CMOS type random access semiconductor memory."

Hillis, 3:42-48.

> "As shall be described in more detail hereinafter, upon system initialization (bootstrapping), an image of the BIOS ROM is copied to the upper 128 kbyte region of system memory, or shadow RAM, from Where

system execution takes place for higher operating speed." Hillis: 6:1-4. *See also* Hillis, 6:16-49.

6.4 to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory,	Hillis, as evidenced by the example citations below, discloses "to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1.1 above.	

6.5 to access the loaded portion of the boot data in the compressed form,	Hillis, as evidenced by the example citations below, discloses "to access the loaded portion of the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to access the loaded portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1.2 above.	

6.6 to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data, and	Hillis, as evidenced by the example citations below, discloses "to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1.3 above.	

6.7 to update the boot data list.	Hillis, as evidenced by the example citations below, discloses "to update the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to update the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1.4.1 above.	

8 (Preamble) A method of loading an operating system for booting a computer system, comprising:	Hillis, as evidenced by the example citations below, discloses "A method of loading an operating system for booting a computer system, comprising:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method of loading an operating system for booting a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1 (Preamble) above.	

8.1 storing a portion of the operating system in a compressed form in a first memory;	Hillis, as evidenced by the example citations below, discloses "storing a portion of the operating system in a compressed form in a first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing a portion of the operating system in a compressed form in a first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 6.2 above.	

8.2 loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list;	Hillis, as evidenced by the example citations below, discloses "loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1.1 above.	

8.3 accessing the loaded portion of the operating system from the second memory in the compressed form;	Hillis, as evidenced by the example citations below, discloses "accessing the loaded portion of the operating system from the second memory in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded portion of the operating system from the second memory in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1.2 above.	

8.4 decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system;	Hillis, as evidenced by the example citations below, discloses "decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 1.3 and 1.4.2 above.	

8.5 utilizing the decompressed portion of the operating system to at least partially boot the computer system; and	Hillis, as evidenced by the example citations below, discloses "utilizing the decompressed portion of the operating system to at least partially boot the computer system"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed portion of the operating system to at least partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hillis discloses this limitation:

See Claims 1.3 and 1.4.2 above.

See also

> "A further advantage is in performing BIOS code decompression under different boot sceneries, cold and warm, and upon memory conditions of real and protect."

Hillis, 3:21-24.

> "As needed, portions of the BIOS code including POST, Setup (if invoked) and then other BIOS routines are selectively decompressed from the shadow memory to another location of the system memory. Normal execution of POST and BIOS then proceeds until the boot is completed."

Hillis, 3:58-63.

> "As BIOS routines are needed, they are now selectively decompressed from system RAM to system RAM (step 56). The POST may be decompressed into any other region of RAM within or outside the system RAM range of addresses including the region ultimately to be occupied by the operating system."

Hillis, 6:55-60.

> "Decompression occurs selectively in a similar manner for other compressed images of the BIOS code, other than setup which in this example has not yet been called (step 60). Control of the system is

> transferred to the shadow image of POST (step 62)."

Hillis, 6:63-67.

> "As has been described, this invention enables compression of most of the contents of the BIOS ROM and boot strapping of the system upon BIOS code decompression, by storing only the initial portion of the POST code in BIOS ROM, sufficient to enable the system memory."

Hillis, 7:66-8:3.

8.6 updating the boot data list,	Hillis, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1.4.1 above.	

8.7 wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form.	Hillis, as evidenced by the example citations below, discloses "wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 1-1.3 and 5.7 above.	

9.1 The method of claim 8, further comprising: compressing an additional portion of the operating system that is not associated with the boot data list; and	Hillis, as evidenced by the example citations below, discloses "the method of claim 8, further comprising: compressing an additional portion of the operating system that is not associated with the boot data list"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, compressing an additional portion of the operating system that is not associated with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hillis discloses this limitation:

See Claim 1.1 above.

See also

> "To increase the effective capacity of BIOS, an initial portion of the power on system reset (POST) code that is required to enable the system memory is stored in ROM in uncompressed form, and substantially the remaining portion of the BIOS code is stored in compressed form."

Hillis, Abstract.

> "After locating a disk with a valid boot record, the BIOS program reads the data stored on the first sector of the disk, and copies that data to specific locations in RAM. This information, found in the same location on every formatted disk, constitutes the DOS boot record. The BIOS then passes control to the boot record which instructs the PC on how to load the two hidden operating system files to RAM (the files named IBMBIO.COM and IBMDOS.COM on IBM computers). After loading other operating system files into RAM to carry out the rest of the boot up sequence, the boot record is no longer needed."

Hillis, 1:46-56.

> "In accordance with an important aspect of the invention, an initial portion of the BIOS code that is required to enable the system memory is in uncompressed form and a remaining portion thereof for carrying out prescribed functions including converting operating signals developed by

an operating system executed by the CPU into electrical signals compatible with devices that are responsive to signals provided by the CPU to the system bus, is in compressed form." Hillis, 3:36-43. "The next layers of the BIOS ROM consist of compressed set-up data, including descriptive text, and compressed setup code. Next, all but the initial portion of the power on system test (POST) code (termed "phase 2 POST herein") is stored in compressed form, the initial portion of POST (termed "phase 1 POST") being stored in the next lower layer of BIOS ROM." Hillis, 5:35-41. "As shall be described in more detail hereinafter, upon system initialization (bootstrapping), an image of the BIOS ROM is copied to the upper 128 kbyte region of system memory, or shadow RAM, from where system execution takes place for higher operating speed. Shadowing of the BIOS is well known. In accordance with the invention, however, most of the BIOS code is stored in ROM in compressed form." Hillis, 6:1-8. "As has been described, this invention enables compression of most of the contents of the BIOS ROM and boot strapping of the system upon BIOS code decompression, by storing only the initial portion of the POST code in BIOS ROM, sufficient to enable the system memory." Hillis, 7:66-8:3.

9.2 storing the additional portion of the operating system in the first memory, and	Hillis, as evidenced by the example citations below, discloses “storing the additional portion of the operating system in the first memory”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing the additional portion of the operating system in the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Hillis discloses this limitation: *See* Claim 8.1 above.	

9.3 wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system.	Hillis, as evidenced by the example citations below, discloses "wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 8.5 above.	

10. The method of claim 9, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder.	Hillis, as evidenced by the example citations below, discloses "the method of claim 9, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hillis discloses this limitation:

See Claim 9.1 above.

See also

> "To increase the effective capacity of BIOS, an initial portion of the power on system reset (POST) code that is required to enable the system memory is stored in ROM in uncompressed form, and substantially the remaining portion of the BIOS code is stored in compressed form."

Hillis, Abstract.

> "After locating a disk with a valid boot record, the BIOS program reads the data stored on the first sector of the disk, and copies that data to specific locations in RAM. This information, found in the same location on every formatted disk, constitutes the DOS boot record. The BIOS then passes control to the boot record which instructs the PC on how to load the two hidden operating system files to RAM (the files named IBMBIO.COM and IBMDOS.COM on IBM computers). After loading other operating system files into RAM to carry out the rest of the boot up sequence, the boot record is no longer needed."

Hillis, 1:46-56.

> "In accordance with an important aspect of the invention, an initial portion of the BIOS code that is required to enable the system memory is in uncompressed form and a remaining portion thereof for carrying out prescribed functions including converting operating signals developed by an operating system executed by the CPU into electrical signals

> compatible with devices that are responsive to signals provided by the CPU to the system bus, is in compressed form."

Hillis, 3:36-43.

> "The next layers of the BIOS ROM consist of compressed set-up data, including descriptive text, and compressed setup code. Next, all but the initial portion of the power on system test (POST) code (termed "phase 2 POST herein") is stored in compressed form, the initial portion of POST (termed "phase 1 POST") being stored in the next lower layer of BIOS ROM."

Hillis, 5:35-41.

> "As shall be described in more detail hereinafter, upon system initialization (bootstrapping), an image of the BIOS ROM is copied to the upper 128 kbyte region of system memory, or shadow RAM, from where system execution takes place for higher operating speed. Shadowing of the BIOS is well known. In accordance with the invention, however, most of the BIOS code is stored in ROM in compressed form."

Hillis, 6:1-8.

> "As has been described, this invention enables compression of most of the contents of the BIOS ROM and boot strapping of the system upon BIOS code decompression, by storing only the initial portion of the POST code in BIOS ROM, sufficient to enable the system memory."

Hillis, 7:66-8:3.

11 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	Hillis, as evidenced by the example citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, comprising:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1 (Preamble) above.	

11.1 loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system;	Hillis, as evidenced by the example citations below, discloses "loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1.1 above. *See also* > "Upon system initialization during a cold boot, the uncompressed portion of POST is executed from the ROM to enable the system memory, and then an image of the BIOS code is written to shadow memory." Hillis, Abstract. > "In accordance with an aspect of the invention, the portion of the BIOS code that is uncompressed in ROM includes an initial portion of a power on system test (POST) code which is sufficient to enable the system memory, a remaining portion of which is compressed." Hillis, 3:44-48. > "Upon cold boot, the initial portion POST is read directly from ROM to enable the system memory, and then an image of the entire BIOS code, the major portion of which is in compressed form, is written to RAM in the system memory, and control is transferred to the image." Hillis, 3:54-58. > "The phase 1 POST code, which is stored in uncompressed (unpacked) form, consists of only that portion of POST that is necessary to enable the system memory. That is, under conventional BIOS protocol, the initial portion of POST is first read from the BIOS ROM to enable or "wake up"	

> the system memory, usually composed of CMOS type random access semiconductor memory."

Hillis, 3:42-48.

> "As shall be described in more detail hereinafter, upon system initialization (bootstrapping), an image of the BIOS ROM is copied to the upper 128 kbyte region of system memory, or shadow RAM, from Where system execution takes place for higher operating speed."

Hillis: 6:1-4.

See also Hillis, 6:16-49.

11.2 accessing the loaded boot data in compressed form from the memory;	Hillis, as evidenced by the example citations below, discloses "accessing the loaded boot data in compressed form from the memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded boot data in compressed form from the memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1.2 above.	

11.3 decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form;	Hillis, as evidenced by the example citations below, discloses "decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1.3 above.	

11.4 utilizing the decompressed boot data to load at least a portion of the operating system for the computer system; and	Hillis, as evidenced by the example citations below, discloses "utilizing the decompressed boot data to load at least a portion of the operating system for the computer system"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed boot data to load at least a portion of the operating system for the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hillis discloses this limitation:

> "A further advantage is in performing BIOS code decompression under different boot sceneries, cold and warm, and upon memory conditions of real and protect."

Hillis, 3:21-24.

> "As needed, portions of the BIOS code including POST, Setup (if invoked) and then other BIOS routines are selectively decompressed from the shadow memory to another location of the system memory. Normal execution of POST and BIOS then proceeds until the boot is completed."

Hillis, 3:58-63.

> "As BIOS routines are needed, they are now selectively decompressed from system RAM to system RAM (step 56). The POST may be decompressed into any other region of RAM within or outside the system RAM range of addresses including the region ultimately to be occupied by the operating system."

Hillis, 6:55-60.

> "Decompression occurs selectively in a similar manner for other compressed images of the BIOS code, other than setup which in this example has not yet been called (step 60). Control of the system is transferred to the shadow image of POST (step 62)."

Hillis, 6:63-67.

> "As has been described, this invention enables compression of most of

the contents of the BIOS ROM and boot strapping of the system upon BIOS code decompression, by storing only the initial portion of the POST code in BIOS ROM, sufficient to enable the system memory." Hillis, 7:66-8:3.

11.5 updating the boot data list.	Hillis, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1.4.1 above.	

Appendix C6
Invalidity of U.S. Patent 8,880,862 based on Hillis

13 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	Hillis, as evidenced by the example citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, comprising"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1 (Preamble) above.	

13.1 loading boot data in a compressed form that is associated with a boot data list from a boot device;	Hillis, as evidenced by the example citations below, discloses "loading boot data in a compressed form that is associated with a boot data list from a boot device"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading boot data in a compressed form that is associated with a boot data list from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1.1 above.	

13.2 accessing the loaded boot data in the compressed form;	Hillis, as evidenced by the example citations below, discloses "accessing the loaded boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1.2 above.	

13.3 decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form;	Hillis, as evidenced by the example citations below, discloses "decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1.3 above.	

13.4 updating the boot data list.	Hillis, as evidenced by the example citations below, discloses "updating the boot data list"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hillis discloses this limitation:

See Claim 1.4.1 above.

14 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	Hillis, as evidenced by the example citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, comprising"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1 (Preamble) above.	

Appendix C6
Invalidity of U.S. Patent 8,880,862 based on Hillis

14.1 accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list;	Hillis, as evidenced by the example citations below, discloses "accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hillis discloses this limitation:

See Claims 1.1 and 1.2 above.

See also

> "After locating a disk with a valid boot record, the BIOS program reads the data stored on the first sector of the disk, and copies that data to specific locations in RAM. This information, found in the same location on every formatted disk, constitutes the DOS boot record. The BIOS then passes control to the boot record which instructs the PC on how to load the two hidden operating system files to RAM (the files named IBMBIO.COM and IBMDOS.COM on IBM computers). After loading other operating system files into RAM to carry out the rest of the boot up sequence, the boot record is no longer needed."

Hillis, 1:46-56.

14.2 loading the boot data into a memory; and	Hillis, as evidenced by the example citations below, discloses "loading the boot data into a memory"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the boot data into a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hillis discloses this limitation:

See Claim 1.1 above.

14.3 servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form; and	Hillis, as evidenced by the example citations below, discloses "servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hillis discloses this limitation:

See Claims 1.2 and 1.3 above.

See also

> "As BIOS code is needed during the remainder of the boot, the code is selectively decompressed from the shadow memory to another region of the system memory to which control is transferred."

Hillis, Abstract.

> "A further advantage is in performing BIOS code decompression under different boot sceneries, cold and warm, and upon memory conditions of real and protect."

Hillis, 3:21-24.

> "Then, after a jump from one location of the system memory to another, decompression of the code takes place."

Hillis, 3:51-53.

> "As needed, portions of the BIOS code including POST, Setup (if

> invoked) and then other BIOS routines are selectively decompressed from the shadow memory to another location of the system memory. Normal execution of POST and BIOS then proceeds until the boot is completed."

Hillis, 3:58-63.

> "Mapping from the BIOS ROM to the system memory is followed by decompression only of those BIOS routines that are necessary. Referring to FIG. 5 depicting the BIOS decompressed shadow RAM image, the upper 64K of shadow memory contains decompressed BIOS code, the lowest 32K portion of the upper 128 kbyte block contains decompressed phase 2 POST code, decompressed set-up data and code reside at the lowest 128K in system memory, and decompressed video and SES reside as shown."

Hillis, 6:8-16.

> "As BIOS routines are needed, they are now selectively decompressed from system RAM to system RAM (step 56). The POST may be decompressed into any other region of RAM within or outside the system RAM range of addresses including the region ultimately to be occupied by the operating system."

Hillis, 6:55-60.

> "Decompression occurs selectively in a similar manner for other compressed images of the BIOS code, other than setup which in this example has not yet been called (step 60). Control of the system is transferred to the shadow image of POST (step 62). Execution of POST continues."

Hillis, 6:63-7:1.

> "As has been described, this invention enables compression of most of the contents of the BIOS ROM and boot strapping of the system upon BIOS code decompression, by storing only the initial portion of the POST code in BIOS ROM, sufficient to enable the system memory."

Hillis, 7:66-8:3.

14.4 updating the boot data list.	Hillis, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 1.4.1 above.	

15. The method of claim 14, wherein the boot data comprises: a program code associated with the operating system.	Hillis, as evidenced by the example citations below, discloses "the method of claim 14, wherein the boot data comprises: a program code associated with the operating system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the boot data comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 1 (Preamble) and 1.1 above. *See also* "Before a PC can run an operating system, it must load the operating system from a disk to the PC's working memory which is ordinarily random access semiconductor memory(RAM). This is carried out through a process known as "bootstrapping," or more simply, "booting" the PC." Hillis, 1:22-26.	

Appendix C6
Invalidity of U.S. Patent 8,880,862 based on Hillis

16. The method of claim 14, wherein the operating system comprises: a plurality of files.	Hillis, as evidenced by the example citations below, discloses "the method of claim 14, wherein the operating system comprises: a plurality of files."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: "Next loaded from the boot disk into RAM is the file COMMAND.COM which is an operating system file containing, among other functions, fundamental DOS commands used throughout application program execution, and a file named AUTOEXECBAT created by the user and containing a series of DOS batch file commands or program names to be executed by the PC each time the computer is turned on." Hillis, 1:65-2:5.	

17. The method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program.	Hillis, as evidenced by the example citations below, discloses "the method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hillis discloses this limitation:

See Claim 15 above.

See also

> "Next loaded from the boot disk into RAM is the file COMMAND.COM which is an operating system file containing, among other functions, fundamental DOS commands used throughout application program execution, and a file named AUTOEXECBAT created by the user and containing a series of DOS batch file commands or program names to be executed by the PC each time the computer is turned on."

Hillis, 1:65-2:5.

> "This is followed in the same region of memory by the operating system, such as DOS, followed by any application programs."

Hillis, 2:14-16.

19.1 The method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:	Hillis, as evidenced by the example citations below, discloses "the method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 1.1 and 1.2 above.	

19.2 associating the accessed boot data that is not associated with the boot data list to the boot data list.	Hillis, as evidenced by the example citations below, discloses "associating the accessed boot data that is not associated with the boot data list to the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, associating the accessed boot data that is not associated with the boot data list to the boot data list.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 2 above.	

23. The method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files.	Hillis, as evidenced by the example citations below, discloses "the method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 16 above.	

24. The method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system.	Hillis, as evidenced by the example citations below, discloses "the method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system.
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 15 above.	

27. The method of claim 1, wherein the memory comprises: a physical memory.	Hillis, as evidenced by the example citations below, discloses "the method of claim 1, wherein the memory comprises: a physical memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 1.1 and 1.2 above.	

28. The method of claim 1, wherein the operating system comprises: a plurality of files.	Hillis, as evidenced by the example citations below, discloses "the method of claim 1, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 16 above.	

29. The method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program.	Hillis, as evidenced by the example citations below, discloses "the method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 15 and 17 above.	

31. The method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access.	Hillis, as evidenced by the example citations below, discloses "the method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 1.1 and 1.2 above.	

32. The method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form.	Hillis, as evidenced by the example citations below, discloses "the method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: "Compression of set-up data and code, phase 2 POST, video, Smart Energy System (TM) and BIOS code is carried out by any commercially available LZ-1 or LZ-2 algorithm, such as the techniques identified in US. Pat. Nos. 4,701,745 and 5,016,009, incorporated herein by reference. Other suitable compression and decompression algorithms, can be used, however." Hillis, 5:61-67.	

33. The method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form.	Hillis, as evidenced by the example citations below, discloses "the method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 32 above.	

35. The method of claim 5, wherein the compressed boot data represents a plurality of files.	Hillis, as evidenced by the example citations below, discloses "the method of claim 5, wherein the compressed boot data represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the compressed boot data represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 16 and 23 above.	

36. The method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system.	Hillis, as evidenced by the example citations below, discloses "the method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 15 and 17 above.	

39. The method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory.	Hillis, as evidenced by the example citations below, discloses “the method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Hillis discloses this limitation: *See* Claims 1.1 and 1.2 above.	

40. The method of claim 36, wherein the operating system comprises: a plurality of files.	Hillis, as evidenced by the example citations below, discloses "the method of claim 36, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 36, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 16 and 23 above.	

43. The method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access.	Hillis, as evidenced by the example citations below, discloses “the method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Hillis discloses this limitation: *See* Claim 31 above.	

Appendix C6
Invalidity of U.S. Patent 8,880,862 based on Hillis

44. The method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data.	Hillis, as evidenced by the example citations below, discloses "the method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 32 above.	

45. The method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data.	Hillis, as evidenced by the example citations below, discloses "the method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 32 and 33 above.	

47. The system of claim 6, wherein the boot data in the compressed form represents a plurality of files.	Hillis, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 16 and 23 above.	

48. The system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system.	Hillis, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 15, 17 and 24 above.	

49. The system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form.	Hillis, as evidenced by the example citations below, discloses "the system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 10 above.	

50. The system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form.	Hillis, as evidenced by the example citations below, discloses "the system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: "Mapping from the BIOS ROM to the system memory is followed by decompression only of those BIOS routines that are necessary. Referring to FIG. 5 depicting the BIOS decompressed shadow RAM image, the upper 64K of shadow memory contains decompressed BIOS code, the lowest 32K portion of the upper 128 kbyte block contains decompressed phase 2 POST code, decompressed set-up data and code reside at the lowest 128K in system memory, and decompressed video and SES reside as shown." Hillis, 6:8-16.	

51. The system of claim 6, wherein the first memory comprises: a physical memory.	Hillis, as evidenced by the example citations below, discloses "the system of claim 6, wherein the first memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the first memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 27 and 39 above.	

52. The system of claim 6, wherein the boot data the compressed form comprises: a plurality of files.	Hillis, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data the compressed form comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data the compressed form comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 16 and 23 above.	

53. The system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program.	Hillis, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 15, 17 and 24 above.	

59. The method of claim 8, wherein the operating system in the compressed form represents a plurality of files.	Hillis, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 16 and 23 above.	

60. The method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system.	Hillis, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 15, 17 and 24 above.	

63. The method of claim 8, wherein the second memory comprises: a physical memory.	Hillis, as evidenced by the example citations below, discloses "the method of claim 8, wherein the second memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the second memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 27 and 38 above.	

Appendix C6
Invalidity of U.S. Patent 8,880,862 based on Hillis

64. The method of claim 8, wherein the operating system comprises: a plurality of files.	Hillis, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 16 and 23 above.	

65. The method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program.	Hillis, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 15, 17 and 24 above.	

67. The method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access.	Hillis, as evidenced by the example citations below, discloses "the method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 31 above.	

71. The method of claim 11, wherein the boot data in the compressed form represents a plurality of files.	Hillis, as evidenced by the example citations below, discloses "the method of claim 11, wherein the boot data in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 16 and 23 above.	

72. The method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system.	Hillis, as evidenced by the example citations below, discloses “the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Hillis discloses this limitation: *See* Claims 15, 17 and 24 above.	

75. The method of claim 11, wherein the memory comprises: a physical memory.	Hillis, as evidenced by the example citations below, discloses "the method of claim 11, wherein the memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 27 above.	

76. The method of claim 11, wherein the operating system comprises: a plurality of files.	Hillis, as evidenced by the example citations below, discloses "the method of claim 11, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 16 and 23 above.	

77. The method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program.	Hillis, as evidenced by the example citations below, discloses "the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 15, 17 and 24 above.	

79. The method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access.	Hillis, as evidenced by the example citations below, discloses "the method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 31 above.	

80. The method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form.	Hillis, as evidenced by the example citations below, discloses "the method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 32, 33 and 49 above.	

81. The method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form.	Hillis, as evidenced by the example citations below, discloses "the method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 32, 33 and 49 above.	

83. The method of claim 13, wherein the boot data in the compressed form represents a plurality of files.	Hillis, as evidenced by the example citations below, discloses "the method of claim 13, wherein the boot data in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 16 and 23 above.	

84. The method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system.	Hillis, as evidenced by the example citations below, discloses “the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Hillis discloses this limitation: *See* Claims 15, 17 and 24 above.	

87. The method of claim 13, wherein the memory comprises: a physical memory.	Hillis, as evidenced by the example citations below, discloses "the method of claim 13, wherein the memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 27 above.	

88. The method of claim 13, wherein the operating system comprises: a plurality of files.	Hillis, as evidenced by the example citations below, discloses “the method of claim 13, wherein the operating system comprises: a plurality of files”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Hillis discloses this limitation: *See* Claims 16 and 23 above.	

89. The method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program.	Hillis, as evidenced by the example citations below, discloses "the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 15, 17 and 24 above.	

91. The method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access.	Hillis, as evidenced by the example citations below, discloses "the method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 31 above.	

92. The method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data.	Hillis, as evidenced by the example citations below, discloses "the method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 32, 33 and 49 above.	

93. The method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data.	Hillis, as evidenced by the example citations below, discloses "the method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claim 32, 33, and 49 above.	

97.1 The method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list,	Hillis, as evidenced by the example citations below, discloses "the method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 9.1-9.3 and 19 above.	

97.2 and wherein the updating comprises: associating the additional compressed boot data with the boot data list.	Hillis, as evidenced by the example citations below, discloses "and wherein the updating comprises: associating the additional compressed boot data with the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, and wherein the updating comprises: associating the additional compressed boot data with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 1.4.1, 2, 4, 5.6, and 8.6 above.	

98. The method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list.	Hillis, as evidenced by the example citations below, discloses "the method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 1.4.1, 5.6, and 8.6 above.	

107. The method of claim 2, further comprising: storing the updated boot list in a non-volatile memory.	Hillis, as evidenced by the example citations below, discloses "the method of claim 2, further comprising: storing the updated boot list in a non-volatile memory."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 2, further comprising: storing the updated boot list in a non-volatile memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hillis discloses this limitation:

See Claims 1.4.1, 5.6, and 8.6 above.

108. The method of claim 2, further comprising: compressing at least a portion of the additional boot data.	Hillis, as evidenced by the example citations below, discloses "the method of claim 2, further comprising: compressing at least a portion of the additional boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 2, further comprising: compressing at least a portion of the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 1.1, 5, 9.1 and 8 above.	

109. The method of claim 108, further comprising: storing the compressed additional boot data.	Hillis, as evidenced by the example citations below, discloses "the method of claim 108, further comprising: storing the compressed additional boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 108, further comprising: storing the compressed additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hillis discloses this limitation: *See* Claims 5.1, 8.1 and 9.1 above.	

Appendix C7
Invalidity of U.S. Patent 8,880,862 based on Horning

U.S. Patent No. 5,420,998 to Horning ("Horning") invalidates claims 1-6, 8-11, 13-17, 19, 23-24, 27-29, 31-33, 35-36, 39-40, 43-45, 47-53, 59-60, 63-65, 67, 71-72, 75-77, 79-81, 83-84, 87-89, 91-93, 97-98, and 107-109 of United States Patent No. 8,880,862 ("the '862 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art, and/or in combination with the knowledge of a person of ordinary skill.

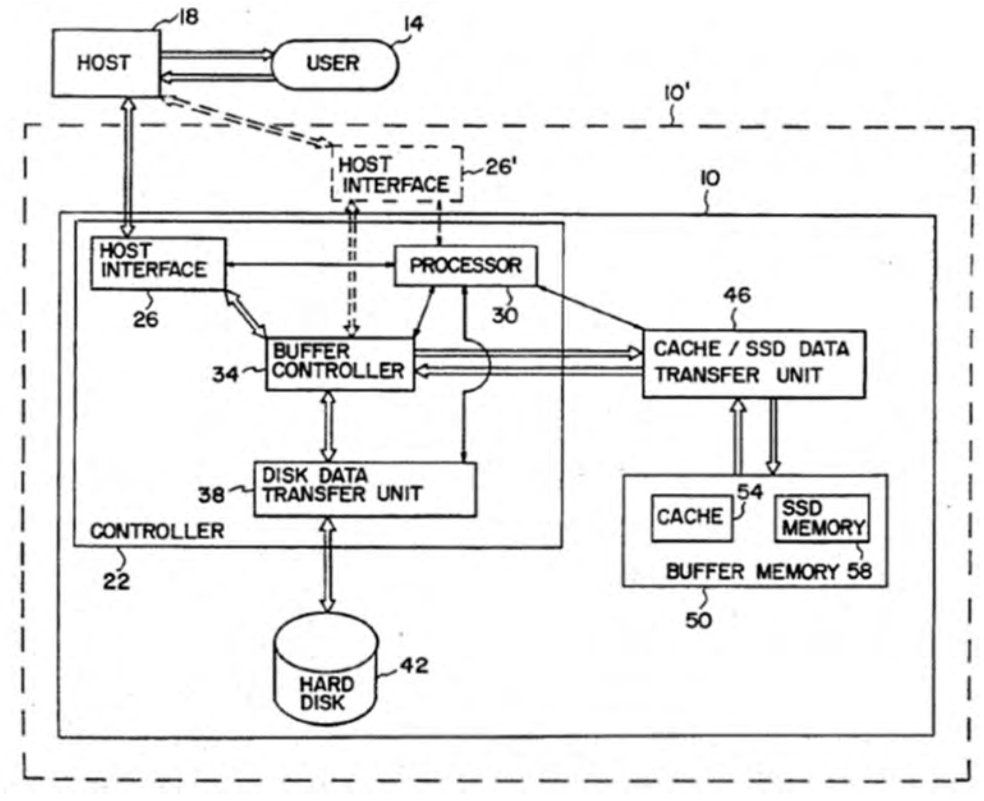
The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '862 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix C7
Invalidity of U.S. Patent 8,880,862 based on Horning

1 (Preamble) A method for providing accelerated loading of an operating system in a computer system, the method comprising:	Horning, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, the method comprising:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system in a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Horning discloses this limitation:



Horning, Fig. 1. *See also* Horning, 2:42-4:15, 4:42-68.

> "Various strategies have been devised to increase the data transfer rate of hard disk technology. One such strategy combines a relatively small volatile solid state data cache together with a hard disk such that data items that are most frequently accessed are maintained in the cache. Since such a cache has a substantially higher data transfer rate than a hard disk, a significant increase in host data processing efficiency can be obtained."

Horning, 1:19-27.

> "In summary, by coalescing a hard disk drive and a solid state disk drive into a single unit, this invention provides a large non-volatile data store and a smaller data store with extremely fast data transfers for data known to be accessed frequently."

Horning, 3:59-63.

> "The hard disk 42 data source provides persistent or nonvolatile data storage together with the necessary data accessing mechanisms and logic for reading and writing data. The buffer memory 50 is preferably solid state memory providing data storage plus data transfer rates much higher than the hard disk 42, in fact, preferably rates greater than the transfer rate of the host 18. Thus, both the cache memory 54 and the SSD memory 58 are intended to supply storage for those data items that are accessed frequently by the host 18. The cache memory 54 stores frequently accessed copies of hard disk 42 data items, preferably without any control bits relating to hard disk 42 data formatting. The SSD memory 58 stores frequently accessed data items in a "disk-like" format."

Horning, 5:17-31.

1.1 loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory;	Horning, as evidenced by the example citations below, discloses "loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Horning discloses this limitation:

> "The hard disk 42 data source provides persistent or nonvolatile data storage together with the necessary data accessing mechanisms and logic for reading and writing data. The buffer memory 50 is preferably solid state memory providing data storage plus data transfer rates much higher than the hard disk 42, in fact, preferably rates greater than the transfer rate of the host 18. Thus, both the cache memory 54 and the SSD memory 58 are intended to supply storage for those data items that are accessed frequently by the host 18. The cache memory 54 stores frequently accessed copies of hard disk 42 data items, preferably without any control bits relating to hard disk 42 data formatting. The SSD memory 58 stores frequently accessed data items in a "disk-like" format."

Horning, 5:17-31.

> "If the data transfer is directed to a hard disk 42 location, then preferably the transfer must be through the cache memory 54 regardless of whether the host request is a read or write. That is, if a data read is requested then a- valid copy of the requested data must either reside in the cache memory 54 or be transferred into the cache memory 54 from the hard disk 42 via the disk data transfer unit 38, the buffer controller 34 and the cache/SSD data transfer unit 46. In any case, the requested data is subsequently transferred from the cache memory 54 to the host 18, via the cache/SSD data transfer unit 46, the buffer controller 34 and the host interface 26."

Horning, 6:33-45.

> "Depending on the dual disk drive 10 configuration, the microcode instructions for the processor 30 can reside in either a read only memory

(ROM) associated with processor 30 or, alternatively, the microcode can reside on the hard disk 42."

Horning, 7:46-50, Fig. 2.

"If the microcode resides in ROM, then the processor 30 can boot-up simultaneously with the hard disk 42 spin-up. Otherwise, the processor 30 boot-up will occur immediately after spin-up and the microcode is downloaded into processor 30."

Horning, 7:50-55.

"In step 68, the processor 30 requests the disk data transfer unit 38 to retrieve the SSD memory 58, or more generally the buffer memory 50, configuration parameter values from a manufacturer specified location on the hard disk 42."

Horning, 7:59-63. *See also* Horning, 7:59-8:2.

"The configuring is similar to step 220 in Fig. 4 in that it includes providing the cache/SSD data transfer unit 46 with: (i) the number of blocks to be read, "blk_cnt," (ii) the initial header location address, in the SSD memory 58 where the data is to be read as determined from the physical address established in step 312 above, (iii) the expected value of the sector header at this location, and (iv) a signal indicating that the data to be received from the SSD memory 58 is to be transferred to the host 18 without header and error correction bits."

Horning, 13:2-12.

"Referring to FIG. 5B, in step 324 the processor 30 configures the host interface 26 and the buffer controller 34 to transfer "blk_" number of data blocks from the cache/SSD data transfer unit 46 to the host 18. In step 328, the processor 30 instructs the host interface 26, the buffer controller 34 and the cache/SSD data transfer unit 46 to transfer data from the SSD memory 58 to the host 19. The requested data is transferred from the SSD memory 58 through the cache/SSD data transfer unit 46 a sector at a time."

Horning, 13:12-22.

"In this step the processor 30 determines whether or not a current version of the data to be read is in the cache 54. If so, then in step 344, the processor assigns the address of the data cache location to the variable, -"cache_loc." In step 348, the processor 30 configures the cache/SSD data transfer unit 46 to read "blk_cnt" data blocks from the

> cache 54 and transfer them to the buffer controller 34 without any modification or formatting. In step 352, the processor 30 configures the host interface 26 and the buffer controller 34 to transfer "blk_cnt" number of data blocks from the cache/SSD data transfer unit 46 to the host 18. Referring to FIG. SC, in step 356, the processor 30 instructs the host interface 26, the buffer controller 34 and the cache/SSD data transfer unit 46 to transfer the data blocks from the cache 54 to the host 18."

Horning, 13:47-62, Fig. 5A-5C.

> "In step 364, the processor 30 determines a cache location where the data can be written from the hard disk 42 and assigns the address of this location to "cache__loc." In step 368, the processor 30 configures the buffer controller 34 and the cache/SSD data transfer unit 46 to transfer "blk_cnt" number of data blocks from the disk data transfer unit 38 to the cache 54. Note, in this step, the cache/SSD data transfer unit 46 is configured to write the data to the cache 54 without any modification or formatting. In step 372, the processor 30 waits for an interrupt from the disk data transfer unit 38 indicating the seek command has completed and successfully located the data to be read. In step 376, the processor instructs the disk transfer unit 38, the buffer controller 34 and the cache/SSD data transfer unit 42 to transfer the data located on the hard disk 42 to the cache 54. Preferably after a predetermined number of data bytes have been transferred to the cache 54, the host interface 26, the buffer controller 34 and the cache/SSD data transfer unit 46 will commence multiplexing the data transfer to the cache 54 with an additional data transfer of this same data from the cache 54 to the host 18."

Horning, 14:1-23, Fig. 5A-5C.

1.2 accessing the loaded portion of the boot data in the compressed form from memory;	Horning, as evidenced by the example citations below, discloses "accessing the loaded portion of the boot data in the compressed form from memory;"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded portion of the boot data in the compressed form from memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: "The hard disk 42 data source provides persistent or nonvolatile data storage together with the necessary data accessing mechanisms and logic for reading and writing data. The buffer memory 50 is preferably solid state memory providing data storage plus data transfer rates much higher than the hard disk 42, in fact, preferably rates greater than the transfer rate of the host 18. Thus, both the cache memory 54 and the SSD memory 58 are intended to supply storage for those data items that are accessed frequently by the host 18. The cache memory 54 stores frequently accessed copies of hard disk 42 data items, preferably without any control bits relating to hard disk 42 data formatting. The SSD memory 58 stores frequently accessed data items in a "disk-like" format." Horning, 5:17-31. "If the data transfer is directed to a hard disk 42 location, then preferably the transfer must be through the cache memory 54 regardless of whether the host request is a read or write. That is, if a data read is requested then a- valid copy of the requested data must either reside in the cache memory 54 or be transferred into the cache memory 54 from the hard disk 42 via the disk data transfer unit 38, the buffer controller 34 and the cache/SSD data transfer unit 46. In any case, the requested data is subsequently transferred from the cache memory 54 to the host 18, via the cache/SSD data transfer unit 46, the buffer controller 34 and the host interface 26." Horning, 6:33-45. "If the microcode resides in ROM, then the processor 30 can boot-up simultaneously with the hard disk 42 spin-up. Otherwise, the processor 30 boot-up will occur immediately after spin-up and the microcode is downloaded into processor 30."	

Horning, 7:50-55.

> "In step 68, the processor 30 requests the disk data transfer unit 38 to retrieve the SSD memory 58, or more generally the buffer memory 50, configuration parameter values from a manufacturer specified location on the hard disk 42."

Horning, 7:59-63. *See also* Horning, 7:59-8:2.

> "Referring to FIG. 5B, in step 324 the processor 30 configures the host interface 26 and the buffer controller 34 to transfer "blk_" number of data blocks from the cache/SSD data transfer unit 46 to the host 18. In step 328, the processor 30 instructs the host interface 26, the buffer controller 34 and the cache/SSD data transfer unit 46 to transfer data from the SSD memory 58 to the host 19. The requested data is transferred from the SSD memory 58 through the cache/SSD data transfer unit 46 a sector at a time."

Horning, 13:12-22.

> "In this step the processor 30 determines whether or not a current version of the data to be read is in the cache 54. If so, then in step 344, the processor assigns the address of the data cache location to the variable, -"cache_loc." In step 348, the processor 30 configures the cache/SSD data transfer unit 46 to read "blk_cnt" data blocks from the cache 54 and transfer them to the buffer controller 34 without any modification or formatting. In step 352, the processor 30 configures the host interface 26 and the buffer controller 34 to transfer "blk_cnt" number of data blocks from the cache/SSD data transfer unit 46 to the host 18. Referring to FIG. SC, in step 356, the processor 30 instructs the host interface 26, the buffer controller 34 and the cache/SSD data transfer unit 46 to transfer the data blocks from the cache 54 to the host 18."

Horning, 13:47-62, Fig. 5A-5C.

> "In step 364, the processor 30 determines a cache location where the data can be written from the hard disk 42 and assigns the address of this location to "cache__loc." In step 368, the processor 30 configures the buffer controller 34 and the cache/SSD data transfer unit 46 to transfer "blk_cnt" number of data blocks from the disk data transfer unit 38 to the cache 54. Note, in this step, the cache/SSD data transfer unit 46 is configured to write the data to the cache 54 without any modification or formatting. In step 372, the processor 30 waits for an interrupt from the disk data transfer unit 38 indicating the seek command has completed and

successfully located the data to be read. In step 376, the processor instructs the disk transfer unit 38, the buffer controller 34 and the cache/SSD data transfer unit 42 to transfer the data located on the hard disk 42 to the cache 54. Preferably after a predetermined number of data bytes have been transferred to the cache 54, the host interface 26, the buffer controller 34 and the cache/SSD data transfer unit 46 will commence multiplexing the data transfer to the cache 54 with an additional data transfer of this same data from the cache 54 to the host 18." Horning, 14:1-23, Fig. 5A-5C.

1.3 decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form; and	Horning, as evidenced by the example citations below, discloses "decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form; and"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

1.4.1 updating the boot data list,	Horning, as evidenced by the example citations below, discloses "updating the boot data list,"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

1.4.2 wherein the decompressed portion of boot data comprises a portion of the operating system.	Horning, as evidenced by the example citations below, discloses "wherein the decompressed portion of boot data comprises a portion of the operating system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the decompressed portion of boot data comprises a portion of the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

2. The method of claim 1, wherein the updating comprises: associating additional boot data with the boot data list.	Horning, as evidenced by the example citations below, discloses "wherein the updating comprises: associating additional boot data with the boot data list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein associating additional boot data with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 1.4.1 above.	

3. The method of claim 1, wherein the updating comprises: removing an association of additional boot data that is associated with the boot data list from the boot data list.	Horning, as evidenced by the example citations below, discloses "wherein the updating comprises: removing an association of additional boot data that is associated with the boot data list from the boot data list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein removing an association of additional boot data that is associated with the boot data list from the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 1.4.1 above.	

4. The method of claim 1, wherein the updating comprises: associating additional boot data with the boot data list; and compressing a portion of the additional boot data.	Horning, as evidenced by the example citations below, discloses "wherein the updating comprises: associating additional boot data with the boot data list; and compressing a portion of the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein associating additional boot data with the boot data list; and compressing a portion of the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 1.4.1, and 2 above.	

Appendix C7
Invalidity of U.S. Patent 8,880,862 based on Horning

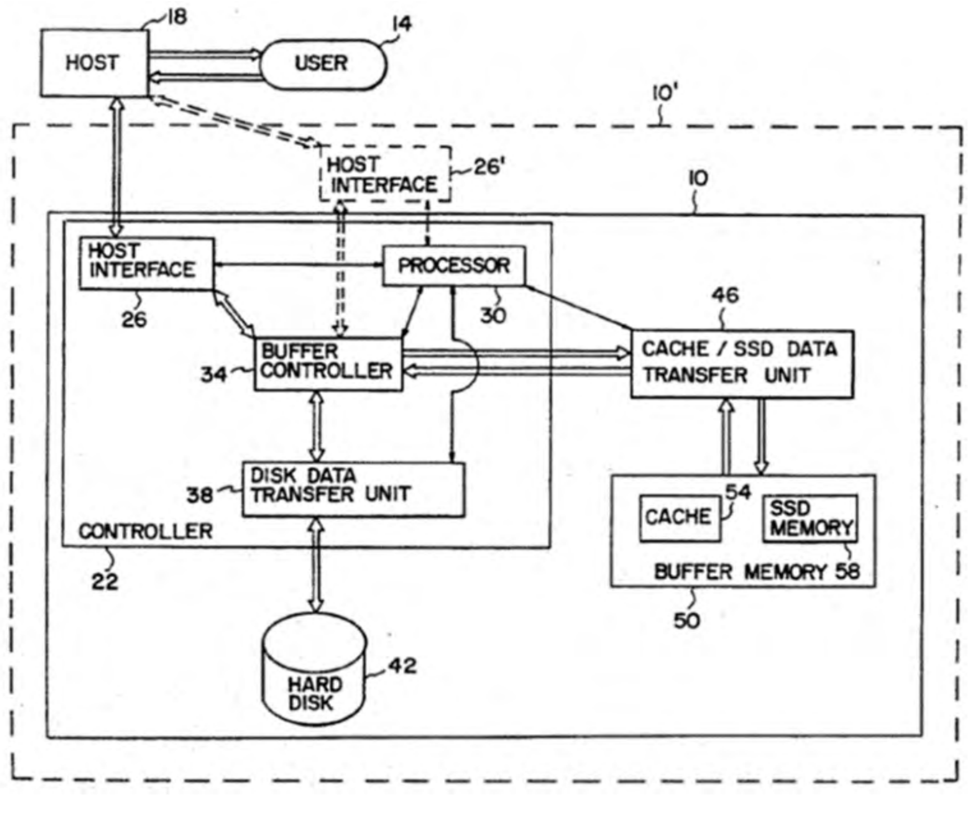
5 (Preamble) A method for booting a computer system, the method comprising:	Horning, as evidenced by the example citations below, discloses "a method for booting a computer system, the method comprising:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Horning discloses this limitation:

See Claims 1-1.4.2 above.

See also



Horning, Fig. 1.

> "The hard disk 42 data source provides persistent or nonvolatile data storage together with the necessary data accessing mechanisms and logic for reading and writing data. The buffer memory 50 is preferably solid state memory providing data storage plus data transfer rates much higher than the hard disk 42, in fact, preferably rates greater than the transfer

> rate of the host 18. Thus, both the cache memory 54 and the SSD memory 58 are intended to supply storage for those data items that are accessed frequently by the host 18. The cache memory 54 stores frequently accessed copies of hard disk 42 data items, preferably without any control bits relating to hard disk 42 data formatting. The SSD memory 58 stores frequently accessed data items in a "disk-like" format."

Horning, 5:17-31.

> "Depending on the dual disk drive 10 configuration, the microcode instructions for the processor 30 can reside in either a read only memory (ROM) associated with processor 30 or, alternatively, the microcode can reside on the hard disk 42."

Horning, 7:46-50, Fig. 2.

> "In step 68, the processor 30 requests the disk data transfer unit 38 to retrieve the SSD memory 58, or more generally the buffer memory 50, configuration parameter values from a manufacturer specified location on the hard disk 42."

Horning, 7:59-63. *See also* Horning, 7:59-8:2.

> "The configuring is similar to step 220 in Fig. 4 in that it includes providing the cache/SSD data transfer unit 46 with: (i) the number of blocks to be read, "blk_cnt," (ii) the initial header location address, in the SSD memory 58 where the data is to be read as determined from the physical address established in step 312 above, (iii) the expected value of the sector header at this location, and (iv) a signal indicating that the data to be received from the SSD memory 58 is to be transferred to the host 18 without header and error correction bits."

Horning, 13:2-12.

> "Additional software is also provided to initialize the dual disk drive and to assure SSD data integrity upon power failure. To make the initialization of the dual disk drive as simple as possible for system administration personnel, the initialization procedure has been constructed so that the dual disk drive can be initialized comparable to a conventional hard disk drive. In this case, the parameters for configuring the dual disk drive are supplied with default values during dual disk drive initialization."

Horning, 3:19-28. *See also*, Horning, 3:28-35.

> "Returning to the controller 22, it includes: a host interface 26, a buffer controller 34, a disk data transfer unit 38 and a processor 30."

Horning, 5:49-52.

> "Referring now to FIG. 2, a preferred procedure is presented for initialization of the dual disk drive 10. In step 60 a user physically switches on power to the dual disk drive 10. Assuming the data connection between the dual disk drive 10 and the host 18 has been physically established, the user preferably activates all further initialization tasks from the host 18. In step 64, both the hard disk 42 and the processor 30 become fully functional. That is, the hard disk 42 is directed to spin-up (i.e. accelerate rotation of its included magnetic storage disk until the proper rotation rate is attained to allow data to be transferred) and the processor 30 is directed to boot-up."

Horning, 7:34-46, Fig. 2.

> "If the microcode resides in ROM, then the processor 30 can boot-up simultaneously with the hard disk 42 spin-up. Otherwise, the processor 30 boot-up will occur immediately after spin-up and the microcode is downloaded into processor 30."

Horning, 7:50-55.

> "Upon completion of the SSD memory 58 formatting, the cache/SSD data transfer unit 46 interrupts the processor 30 signaling that SSD memory 58 initialization is complete."

Horning, 8:16-19.

Appendix C7
Invalidity of U.S. Patent 8,880,862 based on Horning

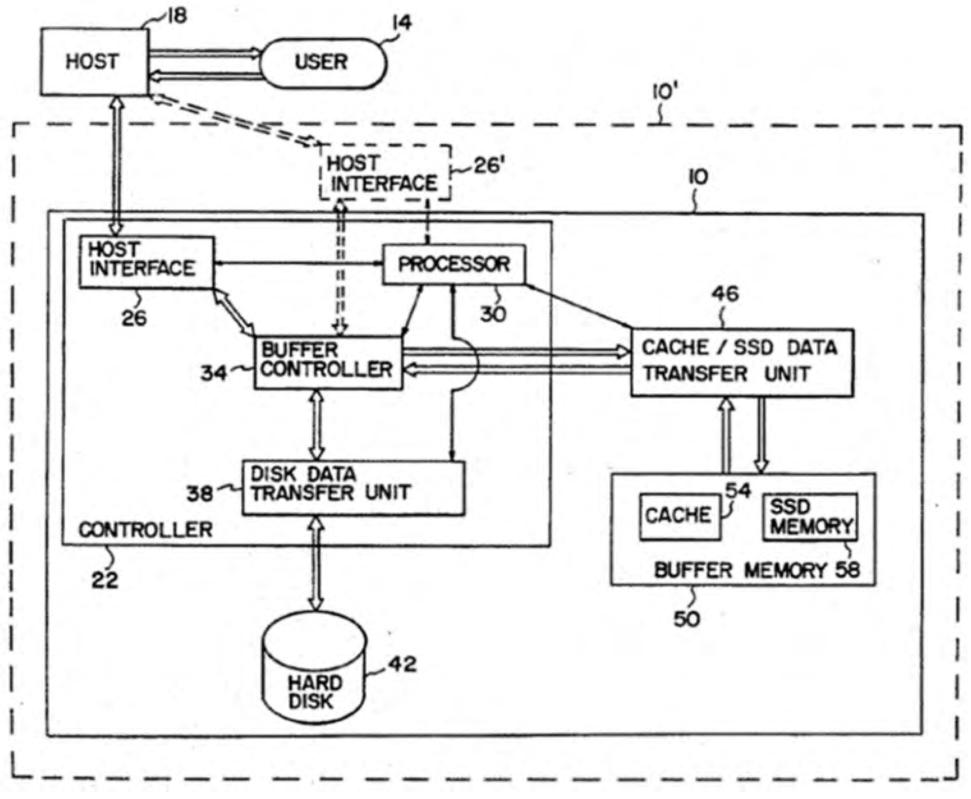
5.1 storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory;	Horning, as evidenced by the example citations below, discloses "storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Horning discloses this limitation:

See Claim 1.1 above.

See also



Horning, Fig. 1.

"The present invention relates to a method and apparatus for storing data such that the storage system produces efficient data transfers with a

> host computational system. The storage system is a dual disk drive system that includes a non-volatile hard disk storage device that has relatively slow data transfer rates in comparison to the host system and a solid state disk (SSD) storage device that has much higher data transfer rates."

Horning, 2:39-46. See also Horning, 2:46-62

> "Depending on the dual disk drive 10 configuration, the microcode instructions for the processor 30 can reside in either a read only memory (ROM) associated with processor 30 or, alternatively, the microcode can reside on the hard disk 42."

Horning, 7:46-50.

5.2 loading the stored compressed boot data from the first memory;	Horning, as evidenced by the example citations below, discloses "loading the stored compressed boot data from the first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the stored compressed boot data from the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 1.1 above.	

5.3 accessing the loaded compressed boot data;	Horning, as evidenced by the example citations below, discloses "accessing the loaded compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 1.2 above.	

5.4 decompressing the accessed compressed boot data;	Horning, as evidenced by the example citations below, discloses "decompressing the accessed compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 1.3 above.	

5.5 utilizing the decompressed boot data to at least partially boot the computer system;	Horning, as evidenced by the example citations below, discloses "utilizing the decompressed boot data to at least partially boot the computer system"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed boot data to at least partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Horning discloses this limitation:

See Claims 1.3 and 1.4.2 above.

See also

> "Referring to FIG. 5B, in step 324 the processor 30 configures the host interface 26 and the buffer controller 34 to transfer "blk_" number of data blocks from the cache/SSD data transfer unit 46 to the host 18. In step 328, the processor 30 instructs the host interface 26, the buffer controller 34 and the cache/SSD data transfer unit 46 to transfer data from the SSD memory 58 to the host 19. The requested data is transferred from the SSD memory 58 through the cache/SSD data transfer unit 46 a sector at a time."

Horning, 13:12-22.

> "In this step the processor 30 determines whether or not a current version of the data to be read is in the cache 54. If so, then in step 344, the processor assigns the address of the data cache location to the variable, -"cache_loc." In step 348, the processor 30 configures the cache/SSD data transfer unit 46 to read "blk_cnt" data blocks from the cache 54 and transfer them to the buffer controller 34 without any modification or formatting. In step 352, the processor 30 configures the host interface 26 and the buffer controller 34 to transfer "blk_cnt" number of data blocks from the cache/SSD data transfer unit 46 to the host 18. Referring to FIG. SC, in step 356, the processor 30 instructs the host interface 26, the buffer controller 34 and the cache/SSD data transfer unit 46 to transfer the data blocks from the cache 54 to the host 18."

Horning, 13:47-62, Fig. 5A-5C.

> "In step 364, the processor 30 determines a cache location where the data

can be written from the hard disk 42 and assigns the address of this location to "cache__loc." In step 368, the processor 30 configures the buffer controller 34 and the cache/SSD data transfer unit 46 to transfer "blk_cnt" number of data blocks from the disk data transfer unit 38 to the cache 54. Note, in this step, the cache/SSD data transfer unit 46 is configured to write the data to the cache 54 without any modification or formatting. In step 372, the processor 30 waits for an interrupt from the disk data transfer unit 38 indicating the seek command has completed and successfully located the data to be read. In step 376, the processor instructs the disk transfer unit 38, the buffer controller 34 and the cache/SSD data transfer unit 42 to transfer the data located on the hard disk 42 to the cache 54. Preferably after a predetermined number of data bytes have been transferred to the cache 54, the host interface 26, the buffer controller 34 and the cache/SSD data transfer unit 46 will commence multiplexing the data transfer to the cache 54 with an additional data transfer of this same data from the cache 54 to the host 18."

Horning, 14:1-23, Fig. 5A-5C.

5.6 updating the boot data list;	Horning, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 1.4.1 above.	

5.7 wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form.	Horning, as evidenced by the example citations below, discloses "wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 1-1.3 above.	

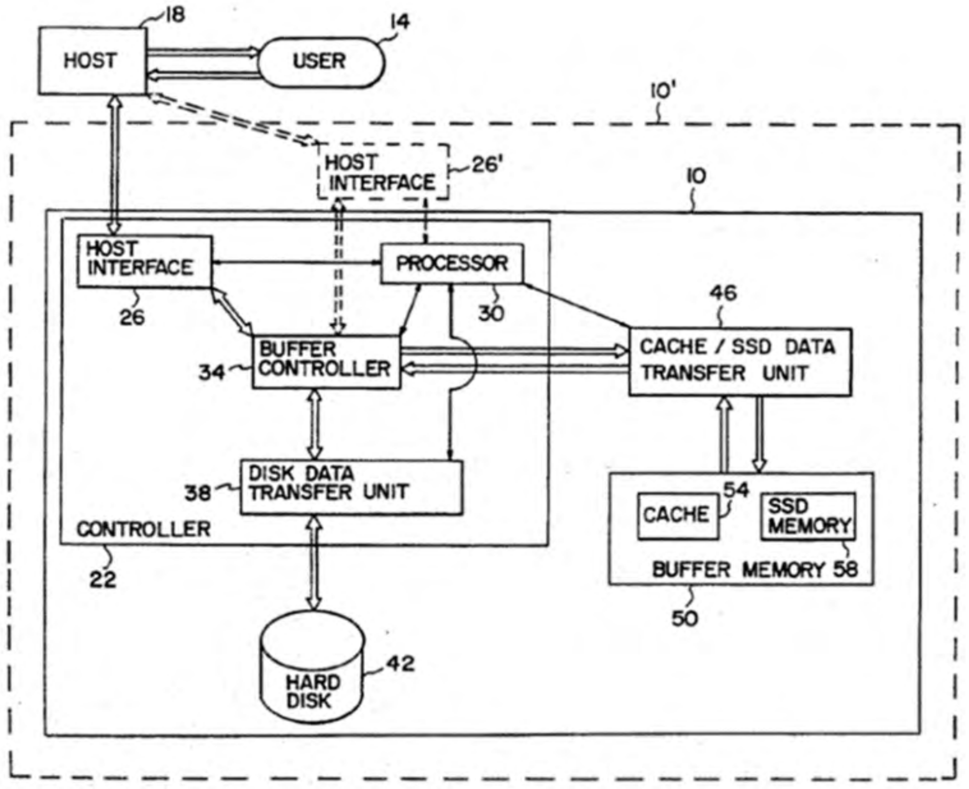
Appendix C7
Invalidity of U.S. Patent 8,880,862 based on Horning

6 (Preamble) A system comprising: a processor;	Horning, as evidenced by the example citations below, discloses "a system comprising: a processor:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system comprising: a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Horning discloses this limitation:

See



Horning, Fig. 1.

> "Additional software is also provided to initialize the dual disk drive and to assure SSD data integrity upon power failure. To make the initialization of the dual disk drive as simple as possible for system administration personnel, the initialization procedure has been constructed so that the dual disk drive can be initialized comparable to a

> conventional hard disk drive. In this case, the parameters for configuring the dual disk drive are supplied with default values during dual disk drive initialization."

Horning, 3:19-28. *See also*, Horning, 3:28-35.

> "Returning to the controller 22, it includes: a host interface 26, a buffer controller 34, a disk data transfer unit 38 and a processor 30."

Horning, 5:49-52.

> "Referring now to FIG. 2, a preferred procedure is presented for initialization of the dual disk drive 10. In step 60 a user physically switches on power to the dual disk drive 10. Assuming the data connection between the dual disk drive 10 and the host 18 has been physically established, the user preferably activates all further initialization tasks from the host 18. In step 64, both the hard disk 42 and the processor 30 become fully functional. That is, the hard disk 42 is directed to spin-up (i.e. accelerate rotation of its included magnetic storage disk until the proper rotation rate is attained to allow data to be transferred) and the processor 30 is directed to boot-up."

Horning, 7:34-46, Fig. 2.

> "If the microcode resides in ROM, then the processor 30 can boot-up simultaneously with the hard disk 42 spin-up. Otherwise, the processor 30 boot-up will occur immediately after spin-up and the microcode is downloaded into processor 30."

Horning, 7:50-55.

> "Upon completion of the SSD memory 58 formatting, the cache/SSD data transfer unit 46 interrupts the processor 30 signaling that SSD memory 58 initialization is complete."

Horning, 8:16-19.

6.1 a memory; and	Horning, as evidenced by the example citations below, discloses "a memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a memory.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 1.1 and 1.2 above.	

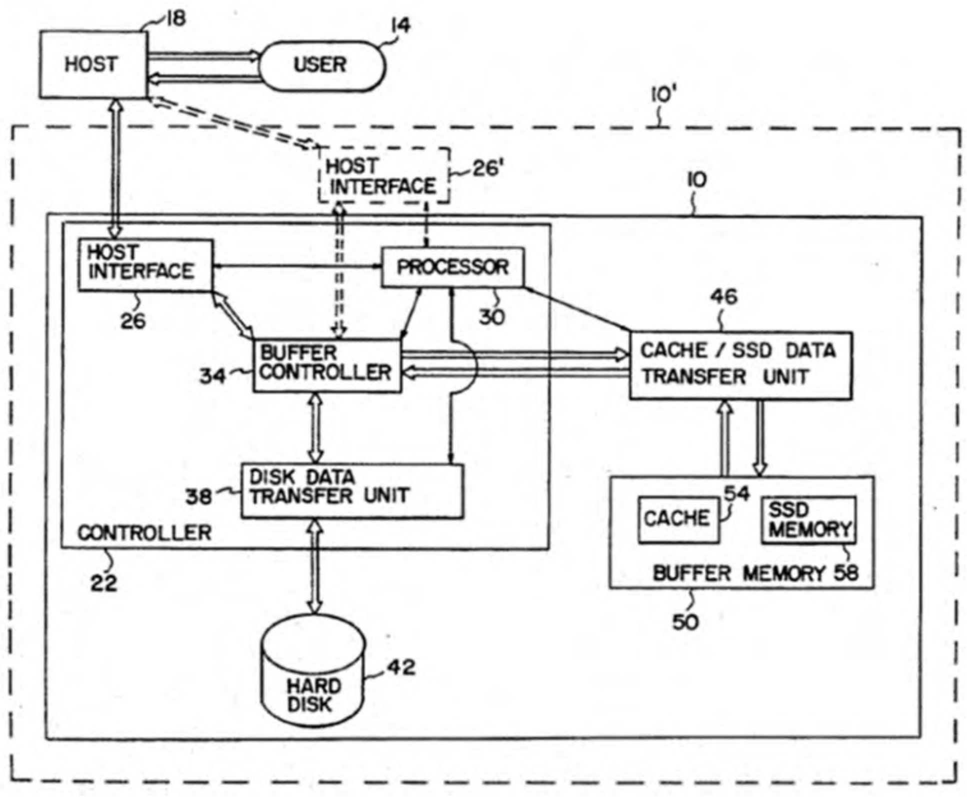
6.2 a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor	Horning, as evidenced by the example citations below, discloses "a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Horning discloses this limitation:

See Claims 1.1 and 1.2 above.

See



Horning, Fig. 1.

> "Additional software is also provided to initialize the dual disk drive and to assure SSD data integrity upon power failure. To make the initialization of the dual disk drive as simple as possible for system administration personnel, the initialization procedure has been constructed so that the dual disk drive can be initialized comparable to a conventional hard disk drive. In this case, the parameters for configuring the dual disk drive are supplied with default values during dual disk drive initialization."

Horning, 3:19-28. *See also*, Horning, 3:28-35.

> "Returning to the controller 22, it includes: a host interface 26, a buffer controller 34, a disk data transfer unit 38 and a processor 30."

Horning, 5:49-52.

> "Referring now to FIG. 2, a preferred procedure is presented for initialization of the dual disk drive 10. In step 60 a user physically switches on power to the dual disk drive 10. Assuming the data connection between the dual disk drive 10 and the host 18 has been physically established, the user preferably activates all further initialization tasks from the host 18. In step 64, both the hard disk 42 and the processor 30 become fully functional. That is, the hard disk 42 is directed to spin-up (i.e. accelerate rotation of its included magnetic storage disk until the proper rotation rate is attained to allow data to be transferred) and the processor 30 is directed to boot-up."

Horning, 7:34-46, Fig. 2.

> "If the microcode resides in ROM, then the processor 30 can boot-up simultaneously with the hard disk 42 spin-up. Otherwise, the processor 30 boot-up will occur immediately after spin-up and the microcode is downloaded into processor 30."

Horning, 7:50-55.

> "Upon completion of the SSD memory 58 formatting, the cache/SSD data transfer unit 46 interrupts the processor 30 signaling that SSD memory 58 initialization is complete."

Horning, 8:16-19.

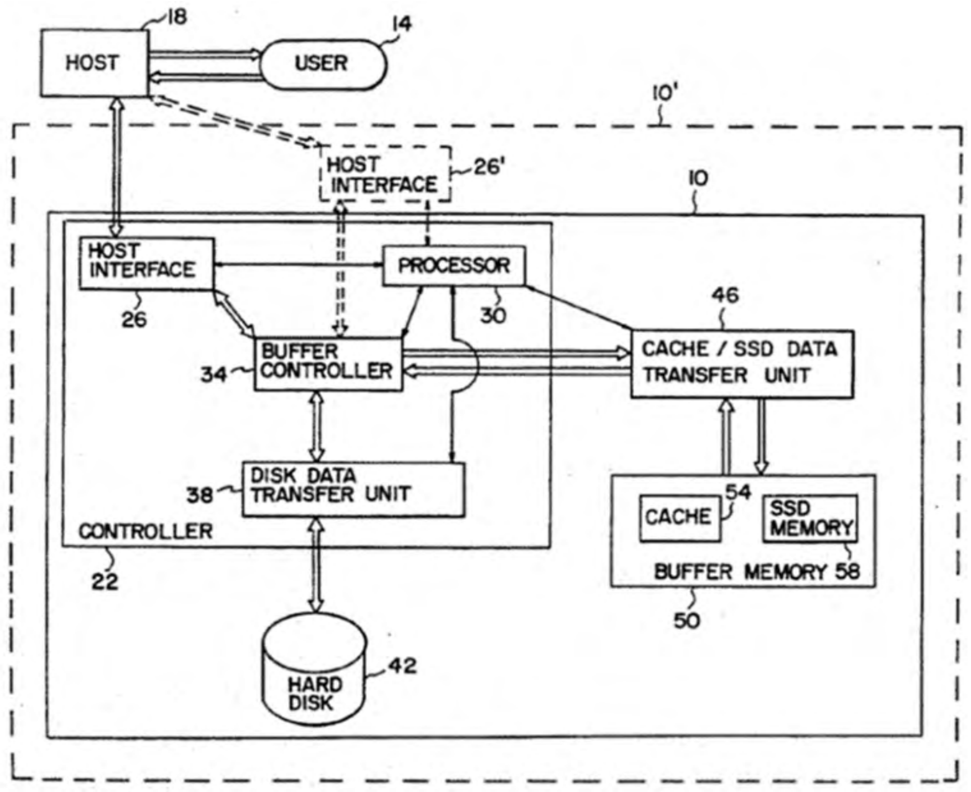
Appendix C7
Invalidity of U.S. Patent 8,880,862 based on Horning

6.3 wherein the processor is configured:	Horning, as evidenced by the example citations below, discloses "wherein the processor is configured:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the processor is configured), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Horning discloses this limitation:

See



Horning, Fig. 1.

> "Additional software is also provided to initialize the dual disk drive and to assure SSD data integrity upon power failure. To make the initialization of the dual disk drive as simple as possible for system administration personnel, the initialization procedure has been constructed so that the dual disk drive can be initialized comparable to a conventional hard disk drive. In this case, the parameters for configuring

> the dual disk drive are supplied with default values during dual disk drive initialization."

Horning, 3:19-28. *See also*, Horning, 3:28-35.

> "Returning to the controller 22, it includes: a host interface 26, a buffer controller 34, a disk data transfer unit 38 and a processor 30."

Horning, 5:49-52.

> "Referring now to FIG. 2, a preferred procedure is presented for initialization of the dual disk drive 10. In step 60 a user physically switches on power to the dual disk drive 10. Assuming the data connection between the dual disk drive 10 and the host 18 has been physically established, the user preferably activates all further initialization tasks from the host 18. In step 64, both the hard disk 42 and the processor 30 become fully functional. That is, the hard disk 42 is directed to spin-up (i.e. accelerate rotation of its included magnetic storage disk until the proper rotation rate is attained to allow data to be transferred) and the processor 30 is directed to boot-up."

Horning, 7:34-46, Fig. 2.

> "If the microcode resides in ROM, then the processor 30 can boot-up simultaneously with the hard disk 42 spin-up. Otherwise, the processor 30 boot-up will occur immediately after spin-up and the microcode is downloaded into processor 30."

Horning, 7:50-55.

> "Upon completion of the SSD memory 58 formatting, the cache/SSD data transfer unit 46 interrupts the processor 30 signaling that SSD memory 58 initialization is complete."

Horning, 8:16-19.

6.4 to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory,	Horning, as evidenced by the example citations below, discloses “to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Horning discloses this limitation: *See* Claim 1.1 above.	

6.5 to access the loaded portion of the boot data in the compressed form,	Horning, as evidenced by the example citations below, discloses "to access the loaded portion of the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to access the loaded portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 1.2 above.	

6.6 to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data, and	Horning, as evidenced by the example citations below, discloses "to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 1.3 above.	

6.7 to update the boot data list.	Horning, as evidenced by the example citations below, discloses “to update the boot data list”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to update the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Horning discloses this limitation: *See* Claim 1.4.1 above.	

Appendix C7
Invalidity of U.S. Patent 8,880,862 based on Horning

8 (Preamble) A method of loading an operating system for booting a computer system, comprising:	Horning, as evidenced by the example citations below, discloses “A method of loading an operating system for booting a computer system, comprising:”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method of loading an operating system for booting a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Horning discloses this limitation: *See* Claim 1 (Preamble) above.	

8.1 storing a portion of the operating system in a compressed form in a first memory;	Horning, as evidenced by the example citations below, discloses "storing a portion of the operating system in a compressed form in a first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing a portion of the operating system in a compressed form in a first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 6.2 above.	

8.2 loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list;	REFERENCE, as evidenced by the example citations below, discloses "loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. REFERENCE discloses this limitation: *See* Claim 1.1 above.	

8.3 accessing the loaded portion of the operating system from the second memory in the compressed form;	Horning, as evidenced by the example citations below, discloses “accessing the loaded portion of the operating system from the second memory in the compressed form”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded portion of the operating system from the second memory in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Horning discloses this limitation: *See* Claim 1.2 above.	

8.4 decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system;	Horning, as evidenced by the example citations below, discloses "decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 1.3 and 1.4.2 above.	

8.5 utilizing the decompressed portion of the operating system to at least partially boot the computer system; and	Horning, as evidenced by the example citations below, discloses "utilizing the decompressed portion of the operating system to at least partially boot the computer system"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed portion of the operating system to at least partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Horning discloses this limitation:

See Claims 1.3 and 1.4.2 above.

See

> "Referring to FIG. 5B, in step 324 the processor 30 configures the host interface 26 and the buffer controller 34 to transfer "blk_" number of data blocks from the cache/SSD data transfer unit 46 to the host 18. In step 328, the processor 30 instructs the host interface 26, the buffer controller 34 and the cache/SSD data transfer unit 46 to transfer data from the SSD memory 58 to the host 19. The requested data is transferred from the SSD memory 58 through the cache/SSD data transfer unit 46 a sector at a time."

Horning, 13:12-22.

> "In this step the processor 30 determines whether or not a current version of the data to be read is in the cache 54. If so, then in step 344, the processor assigns the address of the data cache location to the variable, -"cache_loc." In step 348, the processor 30 configures the cache/SSD data transfer unit 46 to read "blk_cnt" data blocks from the cache 54 and transfer them to the buffer controller 34 without any modification or formatting. In step 352, the processor 30 configures the host interface 26 and the buffer controller 34 to transfer "blk_cnt" number of data blocks from the cache/SSD data transfer unit 46 to the host 18. Referring to FIG. SC, in step 356, the processor 30 instructs the host interface 26, the buffer controller 34 and the cache/SSD data transfer unit 46 to transfer the data blocks from the cache 54 to the host 18."

Horning, 13:47-62, Fig. 5A-5C.

"In step 364, the processor 30 determines a cache location where the data can be written from the hard disk 42 and assigns the address of this location to "cache__loc." In step 368, the processor 30 configures the buffer controller 34 and the cache/SSD data transfer unit 46 to transfer "blk_cnt" number of data blocks from the disk data transfer unit 38 to the cache 54. Note, in this step, the cache/SSD data transfer unit 46 is configured to write the data to the cache 54 without any modification or formatting. In step 372, the processor 30 waits for an interrupt from the disk data transfer unit 38 indicating the seek command has completed and successfully located the data to be read. In step 376, the processor instructs the disk transfer unit 38, the buffer controller 34 and the cache/SSD data transfer unit 42 to transfer the data located on the hard disk 42 to the cache 54. Preferably after a predetermined number of data bytes have been transferred to the cache 54, the host interface 26, the buffer controller 34 and the cache/SSD data transfer unit 46 will commence multiplexing the data transfer to the cache 54 with an additional data transfer of this same data from the cache 54 to the host 18." Horning, 14:1-23, Fig. 5A-5C.

8.6 updating the boot data list,	Horning, as evidenced by the example citations below, discloses “updating the boot data list”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Horning discloses this limitation: *See* Claim 1.4.1 above.	

8.7 wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form.	Horning, as evidenced by the example citations below, discloses "wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 1-1.3 and 5.7 above.	

9.1 The method of claim 8, further comprising: compressing an additional portion of the operating system that is not associated with the boot data list; and	Horning, as evidenced by the example citations below, discloses "the method of claim 8, further comprising: compressing an additional portion of the operating system that is not associated with the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, compressing an additional portion of the operating system that is not associated with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 1.1 above.	

9.2 storing the additional portion of the operating system in the first memory, and	Horning, as evidenced by the example citations below, discloses “storing the additional portion of the operating system in the first memory”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing the additional portion of the operating system in the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Horning discloses this limitation: *See* Claim 8.1 above.	

9.3 wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system.	Horning, as evidenced by the example citations below, discloses "wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 8.5 above.	

10. The method of claim 9, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder.	Horning, as evidenced by the example citations below, discloses "the method of claim 9, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 9.1 above.	

11 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	Horning, as evidenced by the example citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, comprising:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 1 (Preamble) above.	

Appendix C7
Invalidity of U.S. Patent 8,880,862 based on Horning

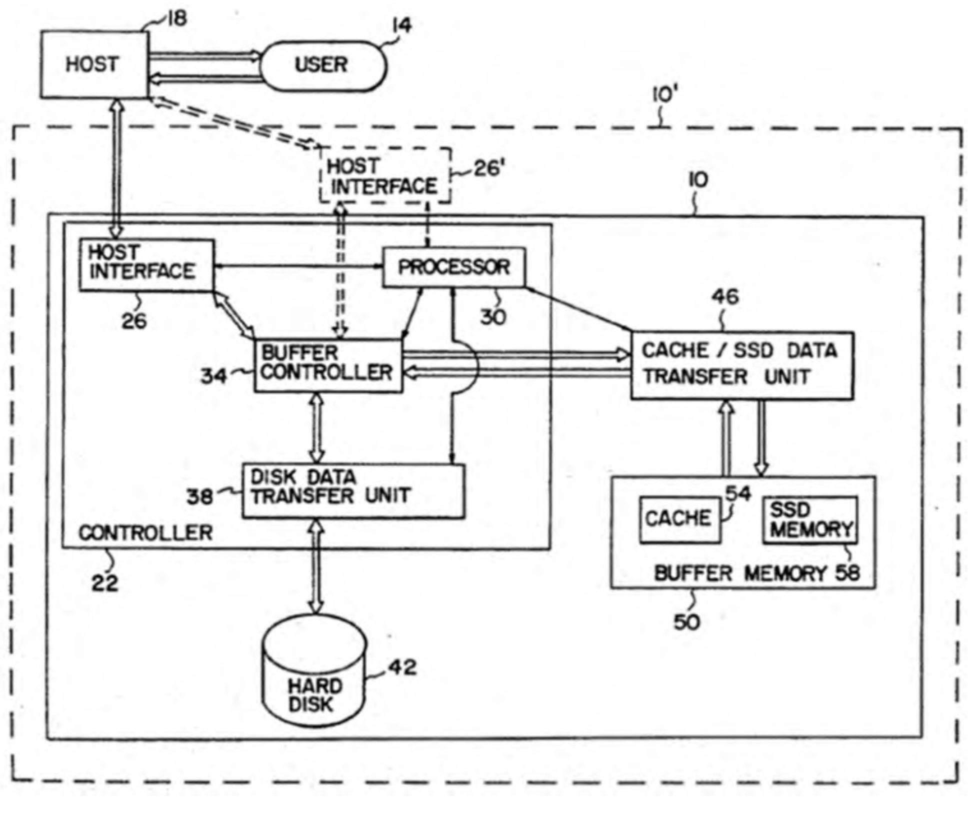
11.1 loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system;	Horning, as evidenced by the example citations below, discloses "loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Horning discloses this limitation:

See Claim 1.1 above.

See also



Horning, Fig. 1

> "Additional software is also provided to initialize the dual disk drive and to assure SSD data integrity upon power failure. To make the initialization of the dual disk drive as simple as possible for system administration personnel, the initialization procedure has been constructed so that the dual disk drive can be initialized comparable to a conventional hard disk drive. In this case, the parameters for configuring the dual disk drive are supplied with default values during dual disk drive initialization."

Horning, 3:19-28. *See also*, Horning, 3:28-35.

> "Returning to the controller 22, it includes: a host interface 26, a buffer controller 34, a disk data transfer unit 38 and a processor 30."

Horning, 5:49-52.

> "Referring now to FIG. 2, a preferred procedure is presented for initialization of the dual disk drive 10. In step 60 a user physically switches on power to the dual disk drive 10. Assuming the data connection between the dual disk drive 10 and the host 18 has been physically established, the user preferably activates all further initialization tasks from the host 18. In step 64, both the hard disk 42 and the processor 30 become fully functional. That is, the hard disk 42 is directed to spin-up (i.e. accelerate rotation of its included magnetic storage disk until the proper rotation rate is attained to allow data to be transferred) and the processor 30 is directed to boot-up."

Horning, 7:34-46, Fig. 2.

> "If the microcode resides in ROM, then the processor 30 can boot-up simultaneously with the hard disk 42 spin-up. Otherwise, the processor 30 boot-up will occur immediately after spin-up and the microcode is downloaded into processor 30."

Horning, 7:50-55.

> "Upon completion of the SSD memory 58 formatting, the cache/SSD data transfer unit 46 interrupts the processor 30 signaling that SSD memory 58 initialization is complete."

Horning, 8:16-19.

11.2 accessing the loaded boot data in compressed form from the memory;	Horning, as evidenced by the example citations below, discloses "accessing the loaded boot data in compressed form from the memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded boot data in compressed form from the memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 1.2 above.	

11.3 decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form;	Horning, as evidenced by the example citations below, discloses "decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 1.3 above.	

11.4 utilizing the decompressed boot data to load at least a portion of the operating system for the computer system; and	Horning, as evidenced by the example citations below, discloses "utilizing the decompressed boot data to load at least a portion of the operating system for the computer system"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed boot data to load at least a portion of the operating system for the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Horning discloses this limitation:

> "Referring to FIG. 5B, in step 324 the processor 30 configures the host interface 26 and the buffer controller 34 to transfer "blk_" number of data blocks from the cache/SSD data transfer unit 46 to the host 18. In step 328, the processor 30 instructs the host interface 26, the buffer controller 34 and the cache/SSD data transfer unit 46 to transfer data from the SSD memory 58 to the host 19. The requested data is transferred from the SSD memory 58 through the cache/SSD data transfer unit 46 a sector at a time."

Horning, 13:12-22.

> "In this step the processor 30 determines whether or not a current version of the data to be read is in the cache 54. If so, then in step 344, the processor assigns the address of the data cache location to the variable, -"cache_loc." In step 348, the processor 30 configures the cache/SSD data transfer unit 46 to read "blk_cnt" data blocks from the cache 54 and transfer them to the buffer controller 34 without any modification or formatting. In step 352, the processor 30 configures the host interface 26 and the buffer controller 34 to transfer "blk_cnt" number of data blocks from the cache/SSD data transfer unit 46 to the host 18. Referring to FIG. SC, in step 356, the processor 30 instructs the host interface 26, the buffer controller 34 and the cache/SSD data transfer unit 46 to transfer the data blocks from the cache 54 to the host 18."

Horning, 13:47-62, Fig. 5A-5C.

> "In step 364, the processor 30 determines a cache location where the data can be written from the hard disk 42 and assigns the address of this location to "cache__loc." In step 368, the processor 30 configures the

buffer controller 34 and the cache/SSD data transfer unit 46 to transfer "blk_cnt" number of data blocks from the disk data transfer unit 38 to the cache 54. Note, in this step, the cache/SSD data transfer unit 46 is configured to write the data to the cache 54 without any modification or formatting. In step 372, the processor 30 waits for an interrupt from the disk data transfer unit 38 indicating the seek command has completed and successfully located the data to be read. In step 376, the processor instructs the disk transfer unit 38, the buffer controller 34 and the cache/SSD data transfer unit 42 to transfer the data located on the hard disk 42 to the cache 54. Preferably after a predetermined number of data bytes have been transferred to the cache 54, the host interface 26, the buffer controller 34 and the cache/SSD data transfer unit 46 will commence multiplexing the data transfer to the cache 54 with an additional data transfer of this same data from the cache 54 to the host 18." Horning, 14:1-23, Fig. 5A-5C.

11.5 updating the boot data list.	Horning, as evidenced by the example citations below, discloses “updating the boot data list”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Horning discloses this limitation: *See* Claim 1.4.1 above.	

13 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	Horning, as evidenced by the example citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, comprising"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 1 (Preamble) above.	

13.1 loading boot data in a compressed form that is associated with a boot data list from a boot device;	Horning, as evidenced by the example citations below, discloses "loading boot data in a compressed form that is associated with a boot data list from a boot device"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading boot data in a compressed form that is associated with a boot data list from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 1.1 above.	

13.2 accessing the loaded boot data in the compressed form;	Horning, as evidenced by the example citations below, discloses "accessing the loaded boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 1.2 above.	

13.3 decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form;	Horning, as evidenced by the example citations below, discloses "decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 1.3 above.	

13.4 updating the boot data list.	Horning, as evidenced by the example citations below, discloses “updating the boot data list”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Horning discloses this limitation: *See* Claim 1.4.1 above.	

Appendix C7
Invalidity of U.S. Patent 8,880,862 based on Horning

14 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	Horning, as evidenced by the example citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, comprising"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 1 (Preamble) above.	

Appendix C7
Invalidity of U.S. Patent 8,880,862 based on Horning

14.1 accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list;	Horning, as evidenced by the example citations below, discloses "accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Horning discloses this limitation:

See Claims 1.1 and 1.2 above.

See also

> "The hard disk 42 data source provides persistent or nonvolatile data storage together with the necessary data accessing mechanisms and logic for reading and writing data. The buffer memory 50 is preferably solid state memory providing data storage plus data transfer rates much higher than the hard disk 42, in fact, preferably rates greater than the transfer rate of the host 18. Thus, both the cache memory 54 and the SSD memory 58 are intended to supply storage for those data items that are accessed frequently by the host 18. The cache memory 54 stores frequently accessed copies of hard disk 42 data items, preferably without any control bits relating to hard disk 42 data formatting. The SSD memory 58 stores frequently accessed data items in a "disk-like" format."

Horning, 5:17-31.

> "Depending on the dual disk drive 10 configuration, the microcode instructions for the processor 30 can reside in either a read only memory (ROM) associated with processor 30 or, alternatively, the microcode can reside on the hard disk 42."

Horning, 7:46-50, Fig. 2.

> "In step 68, the processor 30 requests the disk data transfer unit 38 to retrieve the SSD memory 58, or more generally the buffer memory 50, configuration parameter values from a manufacturer specified location on the hard disk 42."

Horning, 7:59-63. *See also* Horning, 7:59-8:2.

> "The configuring is similar to step 220 in Fig. 4 in that it includes providing the cache/SSD data transfer unit 46 with: (i) the number of blocks to be read, "blk_cnt," (ii) the initial header location address, in the SSD memory 58 where the data is to be read as determined from the physical address established in step 312 above, (iii) the expected value of the sector header at this location, and (iv) a signal indicating that the data to be received from the SSD memory 58 is to be transferred to the host 18 without header and error correction bits."

Horning, 13:2-12.

14.2 loading the boot data into a memory; and	Horning, as evidenced by the example citations below, discloses "loading the boot data into a memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the boot data into a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 1.1 above.	

14.3 servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form; and	Horning, as evidenced by the example citations below, discloses "servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Horning discloses this limitation:

See Claims 1.2 and 1.3 above.

See also

> "The solid state buffer memory, normally used as a data cache in a conventional hard disk drive, in accordance with the present invention, is partitioned into two portions one portion remains as the hard disk data cache as in the conventional hard disk drive configuration while the other portion is used as a memory for an SSD such that the SSD memory is considered by the host to be an entirely separate storage device from that of the hard disk."

Horning, 2:46-55.

> "The hard disk 42 data source provides persistent or nonvolatile data storage together with the necessary data accessing mechanisms and logic for reading and writing data. The buffer memory 50 is preferably solid state memory providing data storage plus data transfer rates much higher than the hard disk 42, in fact, preferably rates greater than the transfer rate of the host 18. Thus, both the cache memory 54 and the SSD memory 58 are intended to supply storage for those data items that are accessed frequently by the host 18. The cache memory 54 stores frequently

accessed copies of hard disk 42 data items, preferably without any control bits relating to hard disk 42 data formatting. The SSD memory 58 stores frequently accessed data items in a "disk-like" format."

Horning, 5:17-31.

"If the data transfer is directed to a hard disk 42 location, then preferably the transfer must be through the cache memory 54 regardless of whether the host request is a read or write. That is, if a data read is requested then a- valid copy of the requested data must either reside in the cache memory 54 or be transferred into the cache memory 54 from the hard disk 42 via the disk data transfer unit 38, the buffer controller 34 and the cache/SSD data transfer unit 46. In any case, the requested data is subsequently transferred from the cache memory 54 to the host 18, via the cache/SSD data transfer unit 46, the buffer controller 34 and the host interface 26."

Horning, 6:33-45.

"Of course, there are well known caching strategies that can be employed to facilitate these data transfers."

Horning, 6:52-54. *See also* Horning, 6:54-7:12.

"If the microcode resides in ROM, then the processor 30 can boot-up simultaneously with the hard disk 42 spin-up. Otherwise, the processor 30 boot-up will occur immediately after spin-up and the microcode is downloaded into processor 30."

Horning, 7:50-55.

"In step 68, the processor 30 requests the disk data transfer unit 38 to retrieve the SSD memory 58, or more generally the buffer memory 50, configuration parameter values from a manufacturer specified location on the hard disk 42."

Horning, 7:59-63. *See also* Horning, 7:59-8:2.

"Referring to FIG. 5B, in step 324 the processor 30 configures the host interface 26 and the buffer controller 34 to transfer "blk_" number of data blocks from the cache/SSD data transfer unit 46 to the host 18. In step 328, the processor 30 instructs the host interface 26, the buffer controller 34 and the cache/SSD data transfer unit 46 to transfer data from the SSD memory 58 to the host 19. The requested data is transferred from the SSD memory 58 through the cache/SSD data transfer unit 46 a sector at a time."

Horning, 13:12-22.

> "In this step the processor 30 determines whether or not a current version of the data to be read is in the cache 54. If so, then in step 344, the processor assigns the address of the data cache location to the variable, -"cache_loc." In step 348, the processor 30 configures the cache/SSD data transfer unit 46 to read "blk_cnt" data blocks from the cache 54 and transfer them to the buffer controller 34 without any modification or formatting. In step 352, the processor 30 configures the host interface 26 and the buffer controller 34 to transfer "blk_cnt" number of data blocks from the cache/SSD data transfer unit 46 to the host 18. Referring to FIG. SC, in step 356, the processor 30 instructs the host interface 26, the buffer controller 34 and the cache/SSD data transfer unit 46 to transfer the data blocks from the cache 54 to the host 18."

Horning, 13:47-62, Fig. 5A-5C.

> "In step 364, the processor 30 determines a cache location where the data can be written from the hard disk 42 and assigns the address of this location to "cache__loc." In step 368, the processor 30 configures the buffer controller 34 and the cache/SSD data transfer unit 46 to transfer "blk_cnt" number of data blocks from the disk data transfer unit 38 to the cache 54. Note, in this step, the cache/SSD data transfer unit 46 is configured to write the data to the cache 54 without any modification or formatting. In step 372, the processor 30 waits for an interrupt from the disk data transfer unit 38 indicating the seek command has completed and successfully located the data to be read. In step 376, the processor instructs the disk transfer unit 38, the buffer controller 34 and the cache/SSD data transfer unit 42 to transfer the data located on the hard disk 42 to the cache 54. Preferably after a predetermined number of data bytes have been transferred to the cache 54, the host interface 26, the buffer controller 34 and the cache/SSD data transfer unit 46 will commence multiplexing the data transfer to the cache 54 with an additional data transfer of this same data from the cache 54 to the host 18."

Horning, 14:1-23, Fig. 5A-5C.

14.4 updating the boot data list.	Horning, as evidenced by the example citations below, discloses “updating the boot data list”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Horning discloses this limitation: *See* Claim 1.4.1 above.	

15. The method of claim 14, wherein the boot data comprises: a program code associated with the operating system.	Horning, as evidenced by the example citations below, discloses "the method of claim 14, wherein the boot data comprises: a program code associated with the operating system."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the boot data comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Horning discloses this limitation:

See Claims 1 (Preamble) and 1.1 above.

16. The method of claim 14, wherein the operating system comprises: a plurality of files.	Horning, as evidenced by the example citations below, discloses “the method of claim 14, wherein the operating system comprises: a plurality of files.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.	

17. The method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program.	Horning, as evidenced by the example citations below, discloses "the method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 15 above.	

19.1 The method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:	Horning, as evidenced by the example citations below, discloses "the method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

19.2 associating the accessed boot data that is not associated with the boot data list to the boot data list.	Horning, as evidenced by the example citations below, discloses "associating the accessed boot data that is not associated with the boot data list to the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, associating the accessed boot data that is not associated with the boot data list to the boot data list.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 2 above.	

Appendix C7
Invalidity of U.S. Patent 8,880,862 based on Horning

23. The method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files.	Horning, as evidenced by the example citations below, discloses "the method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 16 above.	

24. The method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system.	Horning, as evidenced by the example citations below, discloses “the method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system.
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Horning discloses this limitation: *See* Claim 15 above.	

27. The method of claim 1, wherein the memory comprises: a physical memory.	Horning, as evidenced by the example citations below, discloses "the method of claim 1, wherein the memory comprises: a physical memory."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Horning discloses this limitation:

See

> "The hard disk 42 data source provides persistent or nonvolatile data storage together with the necessary data accessing mechanisms and logic for reading and writing data. The buffer memory 50 is preferably solid state memory providing data storage plus data transfer rates much higher than the hard disk 42, in fact, preferably rates greater than the transfer rate of the host 18. Thus, both the cache memory 54 and the SSD memory 58 are intended to supply storage for those data items that are accessed frequently by the host 18. The cache memory 54 stores frequently accessed copies of hard disk 42 data items, preferably without any control bits relating to hard disk 42 data formatting. The SSD memory 58 stores frequently accessed data items in a "disk-like" format."

Horning, 5:17-31.

> "If the data transfer is directed to a hard disk 42 location, then preferably the transfer must be through the cache memory 54 regardless of whether the host request is a read or write. That is, if a data read is requested then a- valid copy of the requested data must either reside in the cache memory 54 or be transferred into the cache memory 54 from the hard disk 42 via the disk data transfer unit 38, the buffer controller 34 and the cache/SSD data transfer unit 46. In any case, the requested data is subsequently transferred from the cache memory 54 to the host 18, via the cache/SSD data transfer unit 46, the buffer controller 34 and the host interface 26."

Horning, 6:33-45.

> "Depending on the dual disk drive 10 configuration, the microcode instructions for the processor 30 can reside in either a read only memory (ROM) associated with processor 30 or, alternatively, the microcode can

> reside on the hard disk 42."

Horning, 7:46-50, Fig. 2.

> "If the microcode resides in ROM, then the processor 30 can boot-up simultaneously with the hard disk 42 spin-up. Otherwise, the processor 30 boot-up will occur immediately after spin-up and the microcode is downloaded into processor 30."

Horning, 7:50-55.

> "In step 364, the processor 30 determines a cache location where the data can be written from the hard disk 42 and assigns the address of this location to "cache__loc." In step 368, the processor 30 configures the buffer controller 34 and the cache/SSD data transfer unit 46 to transfer "blk_cnt" number of data blocks from the disk data transfer unit 38 to the cache 54. Note, in this step, the cache/SSD data transfer unit 46 is configured to write the data to the cache 54 without any modification or formatting. In step 372, the processor 30 waits for an interrupt from the disk data transfer unit 38 indicating the seek command has completed and successfully located the data to be read. In step 376, the processor instructs the disk transfer unit 38, the buffer controller 34 and the cache/SSD data transfer unit 42 to transfer the data located on the hard disk 42 to the cache 54. Preferably after a predetermined number of data bytes have been transferred to the cache 54, the host interface 26, the buffer controller 34 and the cache/SSD data transfer unit 46 will commence multiplexing the data transfer to the cache 54 with an additional data transfer of this same data from the cache 54 to the host 18."

Horning, 14:1-23, Fig. 5A-5C.

Appendix C7
Invalidity of U.S. Patent 8,880,862 based on Horning

28. The method of claim 1, wherein the operating system comprises: a plurality of files.	Horning, as evidenced by the example citations below, discloses "the method of claim 1, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 16 above.	

29. The method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program.	Horning, as evidenced by the example citations below, discloses "the method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 15 and 17 above.	

31. The method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access.	Horning, as evidenced by the example citations below, discloses "the method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 1.1 and 1.2 above.	

32. The method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form.	Horning, as evidenced by the example citations below, discloses "the method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

33. The method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form.	Horning, as evidenced by the example citations below, discloses "the method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

35. The method of claim 5, wherein the compressed boot data represents a plurality of files.	Horning, as evidenced by the example citations below, discloses "the method of claim 5, wherein the compressed boot data represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the compressed boot data represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 16 and 23 above.	

36. The method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system.	Horning, as evidenced by the example citations below, discloses "the method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 15 and 17 above.	

39. The method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory.	Horning, as evidenced by the example citations below, discloses “the method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Horning discloses this limitation: *See* Claims 1.1 and 1.2.	

40. The method of claim 36, wherein the operating system comprises: a plurality of files.	Horning, as evidenced by the example citations below, discloses “the method of claim 36, wherein the operating system comprises: a plurality of files”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 36, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Horning discloses this limitation: *See* Claims 16 and 23 above.	

43. The method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access.	Horning, as evidenced by the example citations below, discloses "the method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 31 above.	

44. The method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data.	Horning, as evidenced by the example citations below, discloses "the method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 32 above.	

45. The method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data.	Horning, as evidenced by the example citations below, discloses "the method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 32 and 33 above.	

47. The system of claim 6, wherein the boot data in the compressed form represents a plurality of files.	Horning, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 16 and 23 above.	

48. The system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system.	Horning, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 15, 17 and 24 above.	

49. The system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form.	Horning, as evidenced by the example citations below, discloses "the system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 10 above.	

50. The system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form.	Horning, as evidenced by the example citations below, discloses "the system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

51. The system of claim 6, wherein the first memory comprises: a physical memory.	Horning, as evidenced by the example citations below, discloses "the system of claim 6, wherein the first memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the first memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 27 and 39 above.	

52. The system of claim 6, wherein the boot data the compressed form comprises: a plurality of files.	Horning, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data the compressed form comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data the compressed form comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 16 and 23 above.	

53. The system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program.	Horning, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 15, 17 and 24 above.	

59. The method of claim 8, wherein the operating system in the compressed form represents a plurality of files.	Horning, as evidenced by the example citations below, discloses “the method of claim 8, wherein the operating system in the compressed form represents a plurality of files”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Horning discloses this limitation: *See* Claims 16 and 23 above.	

60. The method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system.	Horning, as evidenced by the example citations below, discloses “the method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Horning discloses this limitation: *See* Claims 15, 17 and 24 above.	

63. The method of claim 8, wherein the second memory comprises: a physical memory.	Horning, as evidenced by the example citations below, discloses "the method of claim 8, wherein the second memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the second memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 27 and 38 above.	

64. The method of claim 8, wherein the operating system comprises: a plurality of files.	Horning, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 16 and 23 above.	

65. The method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program.	Horning, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 15, 17 and 24 above.	

67. The method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access.	Horning, as evidenced by the example citations below, discloses "the method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 31 above.	

71. The method of claim 11, wherein the boot data in the compressed form represents a plurality of files.	Horning, as evidenced by the example citations below, discloses "the method of claim 11, wherein the boot data in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 16 and 23 above.	

72. The method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system.	Horning, as evidenced by the example citations below, discloses “the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Horning discloses this limitation: *See* Claims 15, 17 and 24 above.	

75. The method of claim 11, wherein the memory comprises: a physical memory.	Horning, as evidenced by the example citations below, discloses "the method of claim 11, wherein the memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 27 above.	

76. The method of claim 11, wherein the operating system comprises: a plurality of files.	Horning, as evidenced by the example citations below, discloses "the method of claim 11, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 16 and 23 above.	

77. The method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program.	Horning, as evidenced by the example citations below, discloses "the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 15, 17 and 24 above.	

79. The method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access.	Horning, as evidenced by the example citations below, discloses "the method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 31 above.	

80. The method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form.	Horning, as evidenced by the example citations below, discloses "the method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 32, 33 and 49 above.	

81. The method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form.	Horning, as evidenced by the example citations below, discloses "the method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 32, 33 and 49 above.	

83. The method of claim 13, wherein the boot data in the compressed form represents a plurality of files.	Horning, as evidenced by the example citations below, discloses "the method of claim 13, wherein the boot data in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 16 and 23 above.	

84. The method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system.	Horning, as evidenced by the example citations below, discloses “the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Horning discloses this limitation: *See* Claims 15, 17 and 24 above.	

87. The method of claim 13, wherein the memory comprises: a physical memory.	Horning, as evidenced by the example citations below, discloses "the method of claim 13, wherein the memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 27 above.	

88. The method of claim 13, wherein the operating system comprises: a plurality of files.	Horning, as evidenced by the example citations below, discloses "the method of claim 13, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 16 and 23 above.	

89. The method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program.	Horning, as evidenced by the example citations below, discloses "the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 15, 17 and 24 above.	

91. The method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access.	Horning, as evidenced by the example citations below, discloses "the method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 31 above.	

92. The method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data.	Horning, as evidenced by the example citations below, discloses "the method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 32, 33 and 49 above.	

93. The method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data.	Horning, as evidenced by the example citations below, discloses "the method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claim 32, 33, and 49 above.	

97.1 The method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list,	Horning, as evidenced by the example citations below, discloses "the method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 9.1-9.3 and 19 above.	

97.2 and wherein the updating comprises: associating the additional compressed boot data with the boot data list.	Horning, as evidenced by the example citations below, discloses "and wherein the updating comprises: associating the additional compressed boot data with the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, and wherein the updating comprises: associating the additional compressed boot data with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 1.4.1, 2, 4, 5.6, and 8.6 above.	

98. The method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list.	Horning, as evidenced by the example citations below, discloses “the method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list”

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.

Horning discloses this limitation:

See Claims 1.4.1, 5.6, and 8.6 above.

107. The method of claim 2, further comprising: storing the updated boot list in a non-volatile memory.	Horning, as evidenced by the example citations below, discloses "the method of claim 2, further comprising: storing the updated boot list in a non-volatile memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 2, further comprising: storing the updated boot list in a non-volatile memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 1.4.1, 5.6, and 8.6 above.	

108. The method of claim 2, further comprising: compressing at least a portion of the additional boot data.	Horning, as evidenced by the example citations below, discloses "the method of claim 2, further comprising: compressing at least a portion of the additional boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 2, further comprising: compressing at least a portion of the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Horning discloses this limitation: *See* Claims 1.1, 5, 9.1 and 8 above.	

109. The method of claim 108, further comprising: storing the compressed additional boot data.	Horning, as evidenced by the example citations below, discloses “the method of claim 108, further comprising: storing the compressed additional boot data”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 108, further comprising: storing the compressed additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Horning discloses this limitation: *See* Claims 5.1, 8.1 and 9.1 above.	

Appendix C8
Invalidity of U.S. Patent 8,880,862 based on Hovis

U.S. Patent No. 5,812,817 to Hovis ("Hovis") invalidates claims 1-6, 8-11, 13-17, 19, 23-24, 27-29, 31-33, 35-36, 39-40, 43-45, 47-53, 59-60, 63-65, 67, 71-72, 75-77, 79-81, 83-84, 87-89, 91-93, 97-98, and 107-109 of United States Patent No. 8,880,862 ("the '862 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

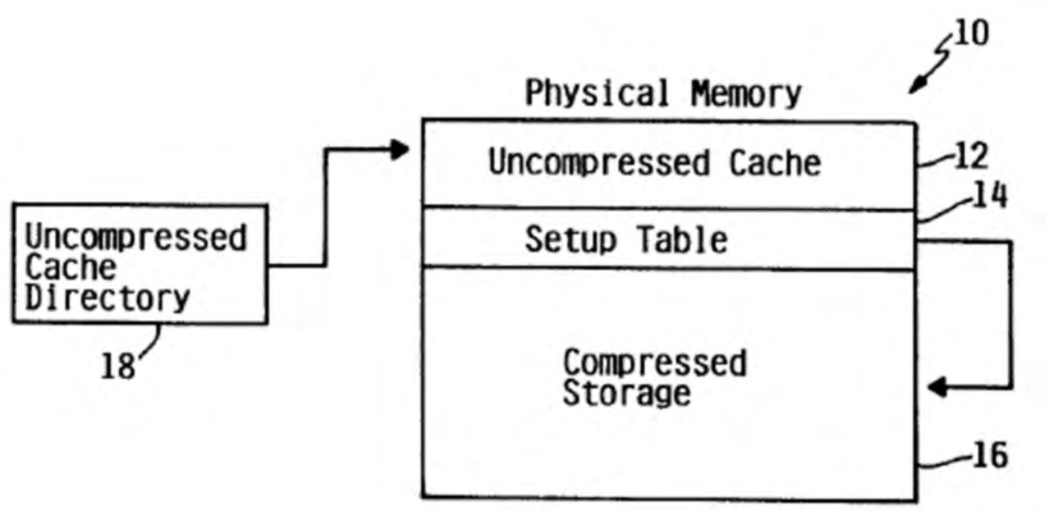
The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '862 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix C8
Invalidity of U.S. Patent 8,880,862 based on Hovis

1 (Preamble) A method for providing accelerated loading of an operating system in a computer system, the method comprising:	Hovis, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, the method comprising:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system in a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hovis discloses this limitation:



Hovis, Fig. 1. Hovis, 1:39-51, 2:18-24, 2:34-43.

> "The function of the setup table 14 is to provide a directory for the memory locations which are in the compressed storage 16. When an access to the memory 10 misses the cache 12, it generates an access to the setup table 14. The data from this access contains the location of the data within compressed storage 16. The address results in an access to the compressed storage 16 which in turn results in compressed data being accessed and processed by the compression engine 28, which performs compression and decompression on the data. The now uncompressed data is placed in the uncompressed cache 12 and transferred to the requesting element (for a fetch), or updated and maintained within the uncompressed cache 12 (for a store)."

Hovis, 3:3-15.

Appendix C8
Invalidity of U.S. Patent 8,880,862 based on Hovis

1.1 loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory;	Hovis, as evidenced by the example citations below, discloses “loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory;”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Hovis discloses this limitation: “The architecture includes a cache section, a setup table, and a compressed storage, all of which are partitioned from a computer memory. The cache section is used for storing uncompressed data and is a fast access memory for data which is frequently referenced. The compressed storage is used for storing compressed data. The setup table is used for specifying locations of compressed data stored within the compressed storage. A high speed uncompressed cache directory is coupled to the memory for determining if data is stored in the cache section or compressed storage and for locating data in the cache.” Hovis, Abstract. Hovis, 1:39-51, 2:18-24, 2:34-43. “The function of the setup table 14 is to provide a directory for the memory locations which are in the compressed storage 16. When an access to the memory 10 misses the cache 12, it generates an access to the setup table 14. The data from this access contains the location of the data within compressed storage 16. The address results in an access to the compressed storage 16 which in turn results in compressed data being accessed and processed by the compression engine 28, which performs compression and decompression on the data. The now uncompressed data is placed in the uncompressed cache 12 and transferred to the requesting element (for a fetch), or updated and maintained within the uncompressed cache 12 (for a store).” Hovis, 3:3-15. *See also* Fig. 2, 3:23-38. “When an access to memory misses the uncompressed cache 12, it results in an access to the setup table 14 in order to fetch pointer information for	

> indicating where the compressed data resides in the compressed storage 16. This data is sent to the compression engine which in turn uses the addresses to fetch the necessary compressed data from compressed storage 16. The data is then uncompressed and sent back to the requesting element and uncompressed cache in the memory by the compression engine 28."

Hovis, 3:63-4:4.

> "Also, the directory for the uncompressed cache must be updated when new locations are added or when old (the least recently used) locations are removed to compressed memory 16."

Hovis, 4:18-21.

> "If the cache was not full, the system uses the setup table to retrieve the compressed data (34) and sends the compressed data to a compression engine for decompressing (38). The system continues as if data was in the cache."

Hovis, 4:40-44.

1.2 accessing the loaded portion of the boot data in the compressed form from memory;	Hovis, as evidenced by the example citations below, discloses "accessing the loaded portion of the boot data in the compressed form from memory;"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded portion of the boot data in the compressed form from memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: "The architecture includes a cache section, a setup table, and a compressed storage, all of which are partitioned from a computer memory. The cache section is used for storing uncompressed data and is a fast access memory for data which is frequently referenced. The compressed storage is used for storing compressed data. The setup table is used for specifying locations of compressed data stored within the compressed storage. A high speed uncompressed cache directory is coupled to the memory for determining if data is stored in the cache section or compressed storage and for locating data in the cache." Hovis, Abstract. Hovis, 1:39-51, 2:18-24, 2:34-43. "The function of the setup table 14 is to provide a directory for the memory locations which are in the compressed storage 16. When an access to the memory 10 misses the cache 12, it generates an access to the setup table 14. The data from this access contains the location of the data within compressed storage 16. The address results in an access to the compressed storage 16 which in turn results in compressed data being accessed and processed by the compression engine 28, which performs compression and decompression on the data. The now uncompressed data is placed in the uncompressed cache 12 and transferred to the requesting element (for a fetch), or updated and maintained within the uncompressed cache 12 (for a store)." Hovis, 3:3-15. "When an access to memory misses the uncompressed cache 12, it results in an access to the setup table 14 in order to fetch pointer information for indicating where the compressed data resides in the compressed storage 16. This data is sent to the compression engine which in turn uses the	

addresses to fetch the necessary compressed data from compressed storage 16. The data is then uncompressed and sent back to the requesting element and uncompressed cache in the memory by the compression engine 28.” Hovis, 3:63-4:4. “If the cache was not full, the system uses the setup table to retrieve the compressed data (34) and sends the compressed data to a compression engine for decompressing (38). The system continues as if data was in the cache.” Hovis, 4:40-44.

1.3 decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form; and	Hovis, as evidenced by the example citations below, discloses "decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hovis discloses this limitation:

> "The architecture includes a cache section, a setup table, and a compressed storage, all of which are partitioned from a computer memory. The cache section is used for storing uncompressed data and is a fast access memory for data which is frequently referenced. The compressed storage is used for storing compressed data. The setup table is used for specifying locations of compressed data stored within the compressed storage. A high speed uncompressed cache directory is coupled to the memory for determining if data is stored in the cache section or compressed storage and for locating data in the cache."

Hovis, Abstract. Hovis, 1:39-51.

> "This invention provides a hardware assisted compression architecture that significantly reduces the typical latency to processor memory as compared to conventional compression techniques. Instead of compressing the entire contents of memory, which adds to the memory latency of all accesses, the present invention reserves a portion of memory as an uncompressed cache storage. Since most memory references are to a relatively small percentage of the stored data, which is preferably stored within the cache, the present invention typically avoids accesses that require additional delay to compress or decompress data while increasing memory capacity."

Hovis, 2:5-16, 2:18-24, 2:34-43.

> "The function of the setup table 14 is to provide a directory for the

> memory locations which are in the compressed storage 16. When an access to the memory 10 misses the cache 12, it generates an access to the setup table 14. The data from this access contains the location of the data within compressed storage 16. The address results in an access to the compressed storage 16 which in turn results in compressed data being accessed and processed by the compression engine 28, which performs compression and decompression on the data. The now uncompressed data is placed in the uncompressed cache 12 and transferred to the requesting element (for a fetch), or updated and maintained within the uncompressed cache 12 (for a store)."

Hovis, 3:3-15.

> "When an access to memory misses the uncompressed cache 12, it results in an access to the setup table 14 in order to fetch pointer information for indicating where the compressed data resides in the compressed storage 16. This data is sent to the compression engine which in turn uses the addresses to fetch the necessary compressed data from compressed storage 16. The data is then uncompressed and sent back to the requesting element and uncompressed cache in the memory by the compression engine 28."

Hovis, 3:63-4:4.

> "If the cache was not full, the system uses the setup table to retrieve the compressed data (34) and sends the compressed data to a compression engine for decompressing (38). The system continues as if data was in the cache."

Hovis, 4:40-44.

1.4.1 updating the boot data list,	Hovis, as evidenced by the example citations below, discloses “updating the boot data list,”

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.

Hovis discloses this limitation:

> “The function of the setup table 14 is to provide a directory for the memory locations which are in the compressed storage 16. When an access to the memory 10 misses the cache 12, it generates an access to the setup table 14. The data from this access contains the location of the data within compressed storage 16. The address results in an access to the compressed storage 16 which in turn results in compressed data being accessed and processed by the compression engine 28, which performs compression and decompression on the data. The now uncompressed data is placed in the uncompressed cache 12 and transferred to the requesting element (for a fetch), or updated and maintained within the uncompressed cache 12 (for a store).”

Hovis, 3:3-15. *See also* Fig. 2, 3:23-38.

> “Also, the directory for the uncompressed cache must be updated when new locations are added or when old (the least recently used) locations are removed to compressed memory 16.”

Hovis, 4:18-21.

1.4.2 wherein the decompressed portion of boot data comprises a portion of the operating system.	Hovis, as evidenced by the example citations below, discloses "wherein the decompressed portion of boot data comprises a portion of the operating system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the decompressed portion of boot data comprises a portion of the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

2. The method of claim 1, wherein the updating comprises: associating additional boot data with the boot data list.	Hovis, as evidenced by the example citations below, discloses "wherein the updating comprises: associating additional boot data with the boot data list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein associating additional boot data with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1.4.1 above.	

3. The method of claim 1, wherein the updating comprises: removing an association of additional boot data that is associated with the boot data list from the boot data list.	Hovis, as evidenced by the example citations below, discloses "wherein the updating comprises: removing an association of additional boot data that is associated with the boot data list from the boot data list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein removing an association of additional boot data that is associated with the boot data list from the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1.4.1 above.	

4. The method of claim 1, wherein the updating comprises: associating additional boot data with the boot data list; and compressing a portion of the additional boot data.	Hovis, as evidenced by the example citations below, discloses "wherein the updating comprises: associating additional boot data with the boot data list; and compressing a portion of the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein associating additional boot data with the boot data list; and compressing a portion of the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 1.4.1, and 2 above.	

5 (Preamble) A method for booting a computer system, the method comprising:	Hovis, as evidenced by the example citations below, discloses "a method for booting a computer system, the method comprising:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hovis discloses this limitation:

See Claims 1-1.4.2 above.

See also

> "The function of the setup table 14 is to provide a directory for the memory locations which are in the compressed storage 16. When an access to the memory 10 misses the cache 12, it generates an access to the setup table 14. The data from this access contains the location of the data within compressed storage 16. The address results in an access to the compressed storage 16 which in turn results in compressed data being accessed and processed by the compression engine 28, which performs compression and decompression on the data. The now uncompressed data is placed in the uncompressed cache 12 and transferred to the requesting element (for a fetch), or updated and maintained within the uncompressed cache 12 (for a store)."

Hovis, 3:3-15. *See also* Fig. 2, 3:23-38.

> "When an access to memory misses the uncompressed cache 12, it results in an access to the setup table 14 in order to fetch pointer information for indicating where the compressed data resides in the compressed storage 16. This data is sent to the compression engine which in turn uses the addresses to fetch the necessary compressed data from compressed storage 16. The data is then uncompressed and sent back to the requesting element and uncompressed cache in the memory by the compression engine 28."

Hovis, 3:63-4:4.

> "Also, the directory for the uncompressed cache must be updated when new locations are added or when old (the least recently used) locations

> are removed to compressed memory 16."

Hovis, 4:18-21.

> "If the cache was not full, the system uses the setup table to retrieve the compressed data (34) and sends the compressed data to a compression engine for decompressing (38). The system continues as if data was in the cache."

Hovis, 4:40-44.

5.1 storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory;	Hovis, as evidenced by the example citations below, discloses "storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hovis discloses this limitation:

See Claim 1.1 above.

See also

> "The setup table is used for specifying locations of compressed data stored within the compressed storage."

Hovis, Abstract. *See also* Hovis, 1:44-45.

> "Since most memory references are to a relatively small percentage of the stored data, which is preferably stored within the cache, the present invention typically avoids accesses that require additional delay to compress or decompress data while increasing memory capacity."

Hovis, 2:11-16.

> "The now uncompressed data is placed in the uncompressed cache 12 and transferred to the requesting element (for a fetch), or updated and maintained within the uncompressed cache 12 (for a store)."

Hovis, 3:12-15.

5.2 loading the stored compressed boot data from the first memory;	Hovis, as evidenced by the example citations below, discloses "loading the stored compressed boot data from the first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the stored compressed boot data from the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1.1 above.	

5.3 accessing the loaded compressed boot data;	Hovis, as evidenced by the example citations below, discloses "accessing the loaded compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1.2 above.	

5.4 decompressing the accessed compressed boot data;	Hovis, as evidenced by the example citations below, discloses "decompressing the accessed compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1.3 above.	

5.5 utilizing the decompressed boot data to at least partially boot the computer system;	Hovis, as evidenced by the example citations below, discloses "utilizing the decompressed boot data to at least partially boot the computer system"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed boot data to at least partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hovis discloses this limitation:

See Claims 1.3 and 1.4.2 above.

See also

> "The function of the setup table 14 is to provide a directory for the memory locations which are in the compressed storage 16. When an access to the memory 10 misses the cache 12, it generates an access to the setup table 14. The data from this access contains the location of the data within compressed storage 16. The address results in an access to the compressed storage 16 which in turn results in compressed data being accessed and processed by the compression engine 28, which performs compression and decompression on the data. The now uncompressed data is placed in the uncompressed cache 12 and transferred to the requesting element (for a fetch), or updated and maintained within the uncompressed cache 12 (for a store)."

Hovis, 3:3-15.

> "When an access to memory misses the uncompressed cache 12, it results in an access to the setup table 14 in order to fetch pointer information for indicating where the compressed data resides in the compressed storage 16. This data is sent to the compression engine which in turn uses the addresses to fetch the necessary compressed data from compressed storage 16. The data is then uncompressed and sent back to the requesting element and uncompressed cache in the memory by the compression engine 28."

Hovis, 3:63-4:4.

> "If the cache was not full, the system uses the setup table to retrieve the compressed data (34) and sends the compressed data to a compression engine for decompressing (38). The system continues as if data was in the

cache.” Hovis, 4:40-44.

5.6 updating the boot data list;	Hovis, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1.4.1 above.	

5.7 wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form.	Hovis, as evidenced by the example citations below, discloses "wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 1-1.3 above.	

Appendix C8
Invalidity of U.S. Patent 8,880,862 based on Hovis

6 (Preamble) A system comprising: a processor;	Hovis, as evidenced by the example citations below, discloses “a system comprising: a processor:”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system comprising: a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.	

6.1 a memory; and	Hovis, as evidenced by the example citations below, discloses "a memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a memory.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 1.1 and 1.2 above.	

6.2 a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor	Hovis, as evidenced by the example citations below, discloses “a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Hovis discloses this limitation: *See* Claims 1.1 and 1.2 above.	

6.3 wherein the processor is configured:	Hovis, as evidenced by the example citations below, discloses "wherein the processor is configured:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the processor is configured), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

6.4 to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory,	Hovis, as evidenced by the example citations below, discloses “to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1.1 above.	

6.5 to access the loaded portion of the boot data in the compressed form,	Hovis, as evidenced by the example citations below, discloses "to access the loaded portion of the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to access the loaded portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1.2 above.	

6.6 to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data, and	Hovis, as evidenced by the example citations below, discloses "to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1.3 above.	

6.7 to update the boot data list.	Hovis, as evidenced by the example citations below, discloses "to update the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to update the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1.4.1 above.	

Appendix C8
Invalidity of U.S. Patent 8,880,862 based on Hovis

8 (Preamble) A method of loading an operating system for booting a computer system, comprising:	Hovis, as evidenced by the example citations below, discloses "A method of loading an operating system for booting a computer system, comprising:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method of loading an operating system for booting a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1 (Preamble) above.	

8.1 storing a portion of the operating system in a compressed form in a first memory;	Hovis, as evidenced by the example citations below, discloses "storing a portion of the operating system in a compressed form in a first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing a portion of the operating system in a compressed form in a first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 6.2 above.	

8.2 loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list;	Hovis, as evidenced by the example citations below, discloses “loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1.1 above.	

8.3 accessing the loaded portion of the operating system from the second memory in the compressed form;	Hovis, as evidenced by the example citations below, discloses "accessing the loaded portion of the operating system from the second memory in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded portion of the operating system from the second memory in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1.2 above.	

8.4 decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system;	Hovis, as evidenced by the example citations below, discloses "decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 1.3 and 1.4.2 above.	

8.5 utilizing the decompressed portion of the operating system to at least partially boot the computer system; and	Hovis, as evidenced by the example citations below, discloses "utilizing the decompressed portion of the operating system to at least partially boot the computer system"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed portion of the operating system to at least partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hovis discloses this limitation:

See Claims 1.3 and 1.4.2 above.

See

> "The function of the setup table 14 is to provide a directory for the memory locations which are in the compressed storage 16. When an access to the memory 10 misses the cache 12, it generates an access to the setup table 14. The data from this access contains the location of the data within compressed storage 16. The address results in an access to the compressed storage 16 which in turn results in compressed data being accessed and processed by the compression engine 28, which performs compression and decompression on the data. The now uncompressed data is placed in the uncompressed cache 12 and transferred to the requesting element (for a fetch), or updated and maintained within the uncompressed cache 12 (for a store)."

Hovis, 3:3-15.

> "When an access to memory misses the uncompressed cache 12, it results in an access to the setup table 14 in order to fetch pointer information for indicating where the compressed data resides in the compressed storage 16. This data is sent to the compression engine which in turn uses the addresses to fetch the necessary compressed data from compressed storage 16. The data is then uncompressed and sent back to the requesting element and uncompressed cache in the memory by the compression engine 28."

Hovis, 3:63-4:4.

> "If the cache was not full, the system uses the setup table to retrieve the

compressed data (34) and sends the compressed data to a compression engine for decompressing (38). The system continues as if data was in the cache." Hovis, 4:40-44.

8.6 updating the boot data list,	Hovis, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1.4.1 above.	

8.7 wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form.	Hovis, as evidenced by the example citations below, discloses “wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Hovis discloses this limitation: *See* Claims 1-1.3 and 5.7 above.	

9.1 The method of claim 8, further comprising: compressing an additional portion of the operating system that is not associated with the boot data list; and	Hovis, as evidenced by the example citations below, discloses "the method of claim 8, further comprising: compressing an additional portion of the operating system that is not associated with the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, compressing an additional portion of the operating system that is not associated with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1.1 above.	

9.2 storing the additional portion of the operating system in the first memory, and	Hovis, as evidenced by the example citations below, discloses “storing the additional portion of the operating system in the first memory”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing the additional portion of the operating system in the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Hovis discloses this limitation: *See* Claim 8.1 above.	

9.3 wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system.	Hovis, as evidenced by the example citations below, discloses “wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Hovis discloses this limitation: *See* Claim 8.5 above.	

Appendix C8
Invalidity of U.S. Patent 8,880,862 based on Hovis

10. The method of claim 9, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder.	Hovis, as evidenced by the example citations below, discloses "the method of claim 9, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hovis discloses this limitation:

See Claim 9.1 above.

> "A compression technique is typically used with this memory architecture. For example, loss-less compression techniques used with the present invention could provide a two times or greater improvement in real memory capacity. This gain is dependent on both data patterns and the chosen compression algorithm. The present invention is not dependent on any particular compression algorithm; it can use any lossless compression algorithm in general."

Hovis, 2:23-30.

> "The address results in an access to the compressed storage 16 Which in turn results in compressed data being accessed and processed by the compression engine 28, Which performs compression and decompression on the data."

Hovis, 3:8-12.

> "FIG. 4 shows a basic data flow structure of the memory 10 with a hardware assist compression engine 28. A memory control 24 interfaces With the memory 10, compression engine 28, and processor units 22. The purpose of a hardware based compression engine 28 is to provide the necessary bandwidth and latency required by memory entities that reside close to the processor data/instruction units (i.e. L1,L2,L3)."

Hovis, 3:41-46

> "The data is then uncompressed and sent back to the requesting element and uncompressed cache in the memory by the compression engine 28."

Hovis, 4:2-4.

> "If the cache is full, then the system must do a cast out and transfer the least recently used data element in the cache to the compressed storage (48), Which requires sending the data to the compression engine for compressing (46). The system then updates the data element in the cache (50) and continues as if the cache Was not full. If the cache Was not full, the system uses the setup table to retrieve the compressed data (34) and sends the compressed data to a compression engine for decompressing (38)."

Hovis, 4:34-44.

> "Aspects of the control of the hardware assisted memory compression can be implemented either by hardware, software (operating system, namely memory management), or a combination of both. The present invention is not dependent upon a particular hardware or software control scheme. The compression engine is preferably implemented in hardware in order to perform the compression and decompression in a timely fashion and With sufficient bandwidth."

Hovis, 5:4-11.

> "The implementation involves a hardware compression engine and the control can be via hardware, software, or a combination of both."

Hovis, 5:27-30.

11 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	Hovis, as evidenced by the example citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, comprising:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1 (Preamble) above.	

11.1 loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system;	Hovis, as evidenced by the example citations below, discloses "loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1.1 above.	

11.2 accessing the loaded boot data in compressed form from the memory;	Hovis, as evidenced by the example citations below, discloses “accessing the loaded boot data in compressed form from the memory”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded boot data in compressed form from the memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1.2 above.	

11.3 decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form;	Hovis, as evidenced by the example citations below, discloses "decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1.3 above.	

11.4 utilizing the decompressed boot data to load at least a portion of the operating system for the computer system; and	Hovis, as evidenced by the example citations below, discloses "utilizing the decompressed boot data to load at least a portion of the operating system for the computer system"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed boot data to load at least a portion of the operating system for the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hovis discloses this limitation:

See

> "The function of the setup table 14 is to provide a directory for the memory locations which are in the compressed storage 16. When an access to the memory 10 misses the cache 12, it generates an access to the setup table 14. The data from this access contains the location of the data within compressed storage 16. The address results in an access to the compressed storage 16 which in turn results in compressed data being accessed and processed by the compression engine 28, which performs compression and decompression on the data. The now uncompressed data is placed in the uncompressed cache 12 and transferred to the requesting element (for a fetch), or updated and maintained within the uncompressed cache 12 (for a store)."

Hovis, 3:3-15.

> "When an access to memory misses the uncompressed cache 12, it results in an access to the setup table 14 in order to fetch pointer information for indicating where the compressed data resides in the compressed storage 16. This data is sent to the compression engine which in turn uses the addresses to fetch the necessary compressed data from compressed storage 16. The data is then uncompressed and sent back to the requesting element and uncompressed cache in the memory by the compression engine 28."

Hovis, 3:63-4:4.

> "If the cache was not full, the system uses the setup table to retrieve the compressed data (34) and sends the compressed data to a compression engine for decompressing (38). The system continues as if data was in the

cache."   Hovis, 4:40-44.

11.5 updating the boot data list.	Hovis, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1.4.1 above.	

Appendix C8
Invalidity of U.S. Patent 8,880,862 based on Hovis

13 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	Hovis, as evidenced by the example citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, comprising"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1 (Preamble) above.	

Appendix C8
Invalidity of U.S. Patent 8,880,862 based on Hovis

13.1 loading boot data in a compressed form that is associated with a boot data list from a boot device;	Hovis, as evidenced by the example citations below, discloses "loading boot data in a compressed form that is associated with a boot data list from a boot device"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading boot data in a compressed form that is associated with a boot data list from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1.1 above.	

13.2 accessing the loaded boot data in the compressed form;	Hovis, as evidenced by the example citations below, discloses “accessing the loaded boot data in the compressed form”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1.2 above.	

13.3 decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form;	Hovis, as evidenced by the example citations below, discloses "decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1.3 above.	

13.4 updating the boot data list.	Hovis, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1.4.1 above.	

Appendix C8
Invalidity of U.S. Patent 8,880,862 based on Hovis

14 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	Hovis, as evidenced by the example citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, comprising"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1 (Preamble) above.	

14.1 accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list;	Hovis, as evidenced by the example citations below, discloses "accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hovis discloses this limitation:

See Claims 1.1 and 1.2 above.

See

> "The function of the setup table 14 is to provide a directory for the memory locations which are in the compressed storage 16. When an access to the memory 10 misses the cache 12, it generates an access to the setup table 14. The data from this access contains the location of the data within compressed storage 16. The address results in an access to the compressed storage 16 which in turn results in compressed data being accessed and processed by the compression engine 28, which performs compression and decompression on the data. The now uncompressed data is placed in the uncompressed cache 12 and transferred to the requesting element (for a fetch), or updated and maintained within the uncompressed cache 12 (for a store)."

Hovis, 3:3-15. *See also* Fig. 2, 3:23-38.

> "When an access to memory misses the uncompressed cache 12, it results in an access to the setup table 14 in order to fetch pointer information for indicating where the compressed data resides in the compressed storage 16. This data is sent to the compression engine which in turn uses the addresses to fetch the necessary compressed data from compressed storage 16. The data is then uncompressed and sent back to the requesting element and uncompressed cache in the memory by the compression engine 28."

Hovis, 3:63-4:4.

> "Also, the directory for the uncompressed cache must be updated when new locations are added or when old (the least recently used) locations are removed to compressed memory 16."

Hovis, 4:18-21.

> "If the cache was not full, the system uses the setup table to retrieve the compressed data (34) and sends the compressed data to a compression engine for decompressing (38). The system continues as if data was in the cache."

Hovis, 4:40-44.

14.2 loading the boot data into a memory; and	Hovis, as evidenced by the example citations below, discloses "loading the boot data into a memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the boot data into a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1.1 above.	

14.3 servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form; and	Hovis, as evidenced by the example citations below, discloses "servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hovis discloses this limitation:

See Claims 1.2 and 1.3 above.

See

> "The architecture includes a cache section, a setup table, and a compressed storage, all of which are partitioned from a computer memory. The cache section is used for storing uncompressed data and is a fast access memory for data which is frequently referenced. The compressed storage is used for storing compressed data. The setup table is used for specifying locations of compressed data stored within the compressed storage. A high speed uncompressed cache directory is coupled to the memory for determining if data is stored in the cache section or compressed storage and for locating data in the cache."

Hovis, Abstract. Hovis, 1:39-51, 2:18-24, 2:34-43.

> "This invention provides a hardware assisted compression architecture that significantly reduces the typical latency to processor memory as compared to conventional compression techniques. Instead of compressing the entire contents of memory, which adds to the memory latency of all accesses, the present invention reserves a portion of memory as an uncompressed cache storage. Since most memory

references are to a relatively small percentage of the stored data, which is preferably stored within the cache, the present invention typically avoids accesses that require additional delay to compress or decompress data while increasing memory capacity."

Hovis, 2:5-16.

"In order to improve performance, the uncompressed cache directory typically has a fast access time. Within the general scheme of processor memory accesses, most memory accesses fall within a small range of the total available memory storage. A memory architecture, according to the present invention, can be used with a most recently used control scheme to maintain the most active segments of memory within the uncompressed storage cache 12. The directory 18 for uncompressed storage preferably must have a short access time in comparison to the overall access time of the uncompressed cache 12 in order to minimally impact typical accesses to memory."

Hovis, 2:44-55.

"Since most accesses will be to uncompressed cache, additional performance degradations associated with going through compression are usually not encountered. This is true even with the proportionately small uncompressed cache."

Hovis, 2:60-63.

"The function of the setup table 14 is to provide a directory for the memory locations which are in the compressed storage 16. When an access to the memory 10 misses the cache 12, it generates an access to the setup table 14. The data from this access contains the location of the data within compressed storage 16. The address results in an access to the compressed storage 16 which in turn results in compressed data being accessed and processed by the compression engine 28, which performs compression and decompression on the data. The now uncompressed data is placed in the uncompressed cache 12 and transferred to the requesting element (for a fetch), or updated and maintained within the uncompressed cache 12 (for a store)."

Hovis, 3:3-15.

"Access latency over conventional systems without compression is typically only degraded by the amount of time it takes to access the

> uncompressed cache directory 18."

Hovis, 3:58-60.

> "When an access to memory misses the uncompressed cache 12, it results in an access to the setup table 14 in order to fetch pointer information for indicating where the compressed data resides in the compressed storage 16. This data is sent to the compression engine which in turn uses the addresses to fetch the necessary compressed data from compressed storage 16. The data is then uncompressed and sent back to the requesting element and uncompressed cache in the memory by the compression engine 28."

Hovis, 3:63-4:4.

> "If the cache was not full, the system uses the setup table to retrieve the compressed data (34) and sends the compressed data to a compression engine for decompressing (38). The system continues as if data was in the cache."

Hovis, 4:40-44.

14.4 updating the boot data list.	Hovis, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 1.4.1 above.	

15. The method of claim 14, wherein the boot data comprises: a program code associated with the operating system.	Hovis, as evidenced by the example citations below, discloses "the method of claim 14, wherein the boot data comprises: a program code associated with the operating system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the boot data comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 1 (Preamble) and 1.1 above.	

16. The method of claim 14, wherein the operating system comprises: a plurality of files.	Hovis, as evidenced by the example citations below, discloses "the method of claim 14, wherein the operating system comprises: a plurality of files."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: "Aspects of the control of the hardware assisted memory compression can be implemented either by hardware, software (operating system, namely memory management), or a combination of both." Hovis, 5:4-7. "This requires an ability within the operating system to support a memory system request to cast data out to I/O when full or near full." Hovis, 17-19.	

17. The method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program.	Hovis, as evidenced by the example citations below, discloses "the method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hovis discloses this limitation:

See Claim 15 above.

19.1 The method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:	Hovis, as evidenced by the example citations below, discloses “the method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.	

19.2 associating the accessed boot data that is not associated with the boot data list to the boot data list.	Hovis, as evidenced by the example citations below, discloses "associating the accessed boot data that is not associated with the boot data list to the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, associating the accessed boot data that is not associated with the boot data list to the boot data list.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 2 above.	

23. The method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files.	Hovis, as evidenced by the example citations below, discloses "the method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 16 above.	

24. The method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system.	Hovis, as evidenced by the example citations below, discloses "the method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system.
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 15 above.	

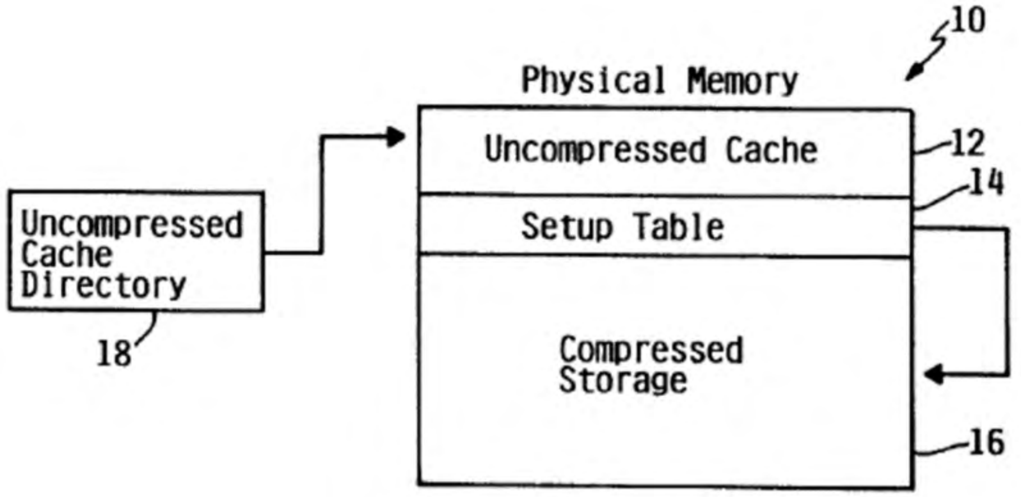
27. The method of claim 1, wherein the memory comprises: a physical memory.	Hovis, as evidenced by the example citations below, discloses "the method of claim 1, wherein the memory comprises: a physical memory."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hovis discloses this limitation:

See ***<u>Add disclosure from Claims 1.1 and 1.2 to the extent a physical memory is disclosed.</u>***

See ***<u>Add reference to physical memory</u>***



Hovis, Fig. 1.

> "The architecture includes a cache section, a setup table, and a compressed storage, all of which are partitioned from a computer memory. The cache section is used for storing uncompressed data and is a fast access memory for data which is frequently referenced. The compressed storage is used for storing compressed data. The setup table is used for specifying locations of compressed data stored within the compressed storage. A high speed uncompressed cache directory is coupled to the memory for determining if data is stored in the cache

> section or compressed storage and for locating data in the cache."

Hovis, Abstract. Hovis, 1:39-51, 2:18-24, 2:34-43.

> "The function of the setup table 14 is to provide a directory for the memory locations which are in the compressed storage 16. When an access to the memory 10 misses the cache 12, it generates an access to the setup table 14. The data from this access contains the location of the data within compressed storage 16. The address results in an access to the compressed storage 16 which in turn results in compressed data being accessed and processed by the compression engine 28, which performs compression and decompression on the data. The now uncompressed data is placed in the uncompressed cache 12 and transferred to the requesting element (for a fetch), or updated and maintained within the uncompressed cache 12 (for a store)."

Hovis, 3:3-15. *See also* Fig. 2, 3:23-38.

> "When an access to memory misses the uncompressed cache 12, it results in an access to the setup table 14 in order to fetch pointer information for indicating where the compressed data resides in the compressed storage 16. This data is sent to the compression engine which in turn uses the addresses to fetch the necessary compressed data from compressed storage 16. The data is then uncompressed and sent back to the requesting element and uncompressed cache in the memory by the compression engine 28."

Hovis, 3:63-4:4.

> "If the cache was not full, the system uses the setup table to retrieve the compressed data (34) and sends the compressed data to a compression engine for decompressing (38). The system continues as if data was in the cache."

Hovis, 4:40-44.

28. The method of claim 1, wherein the operating system comprises: a plurality of files.	Hovis, as evidenced by the example citations below, discloses “the method of claim 1, wherein the operating system comprises: a plurality of files”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Hovis discloses this limitation: *See* Claim 16 above.	

29. The method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program.	Hovis, as evidenced by the example citations below, discloses “the method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Hovis discloses this limitation: *See* Claims 15 and 17 above.	

31. The method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access.	Hovis, as evidenced by the example citations below, discloses "the method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hovis discloses this limitation:

> "The architecture includes a cache section, a setup table, and a compressed storage, all of which are partitioned from a computer memory. The cache section is used for storing uncompressed data and is a fast access memory for data which is frequently referenced. The compressed storage is used for storing compressed data. The setup table is used for specifying locations of compressed data stored within the compressed storage. A high speed uncompressed cache directory is coupled to the memory for determining if data is stored in the cache section or compressed storage and for locating data in the cache."

Hovis, Abstract. Hovis, 1:39-51, 2:18-24, 2:34-43.

> "The function of the setup table 14 is to provide a directory for the memory locations which are in the compressed storage 16. When an access to the memory 10 misses the cache 12, it generates an access to the setup table 14. The data from this access contains the location of the data within compressed storage 16. The address results in an access to the compressed storage 16 which in turn results in compressed data being accessed and processed by the compression engine 28, which performs compression and decompression on the data. The now uncompressed data is placed in the uncompressed cache 12 and transferred to the requesting element (for a fetch), or updated and maintained within the uncompressed cache 12 (for a store)."

Hovis, 3:3-15. *See also* Fig. 2, 3:23-38.

> "When an access to memory misses the uncompressed cache 12, it results in an access to the setup table 14 in order to fetch pointer information for indicating where the compressed data resides in the compressed storage

16. This data is sent to the compression engine which in turn uses the addresses to fetch the necessary compressed data from compressed storage 16. The data is then uncompressed and sent back to the requesting element and uncompressed cache in the memory by the compression engine 28." Hovis, 3:63-4:4. "Also, the directory for the uncompressed cache must be updated when new locations are added or when old (the least recently used) locations are removed to compressed memory 16." Hovis, 4:18-21. "If the cache was not full, the system uses the setup table to retrieve the compressed data (34) and sends the compressed data to a compression engine for decompressing (38). The system continues as if data was in the cache." Hovis, 4:40-44.

32. The method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form.	Hovis, as evidenced by the example citations below, discloses "the method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: "A compression technique is typically used With this memory architecture. For example, loss-less compression techniques used with the present invention could provide a two times or greater improvement in real memory capacity. This gain is dependent on both data patterns and the chosen compression algorithm. The present invention is not dependent on any particular compression algorithm; it can use any lossless compression algorithm in general." Hovis, 2:23-30.	

33. The method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form.	Hovis, as evidenced by the example citations below, discloses "the method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 32 above.	

35. The method of claim 5, wherein the compressed boot data represents a plurality of files.	Hovis, as evidenced by the example citations below, discloses "the method of claim 5, wherein the compressed boot data represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the compressed boot data represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 16 and 23 above.	

36. The method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system.	Hovis, as evidenced by the example citations below, discloses "the method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 15 and 17 above.	

39. The method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory.	Hovis, as evidenced by the example citations below, discloses “the method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Hovis discloses this limitation: *See* Claims 1.1 and 1.2 above.	

40. The method of claim 36, wherein the operating system comprises: a plurality of files.	Hovis, as evidenced by the example citations below, discloses "the method of claim 36, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 36, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 16 and 23 above.	

43. The method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access.	Hovis, as evidenced by the example citations below, discloses "the method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 31 above.	

44. The method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data.	Hovis, as evidenced by the example citations below, discloses “the method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Hovis discloses this limitation: *See* Claim 32 above.	

45. The method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data.	Hovis, as evidenced by the example citations below, discloses "the method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 32 and 33 above.	

47. The system of claim 6, wherein the boot data in the compressed form represents a plurality of files.	Hovis, as evidenced by the example citations below, discloses “the system of claim 6, wherein the boot data in the compressed form represents a plurality of files”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Hovis discloses this limitation: *See* Claims 16 and 23 above.	

48. The system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system.	Hovis, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 15, 17 and 24 above.	

49. The system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form.	Hovis, as evidenced by the example citations below, discloses “the system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Hovis discloses this limitation: *See* Claim 10 above.	

50. The system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form.	Hovis, as evidenced by the example citations below, discloses "the system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hovis discloses this limitation:

See ***<u>Add disclosure where a decoder configured to decompress the boot data in the compressed form</u>***

> "A compression technique is typically used with this memory architecture. For example, loss-less compression techniques used with the present invention could provide a two times or greater improvement in real memory capacity. This gain is dependent on both data patterns and the chosen compression algorithm. The present invention is not dependent on any particular compression algorithm; it can use any lossless compression algorithm in general."

Hovis, 2:23-30.

> "The address results in an access to the compressed storage 16 Which in turn results in compressed data being accessed and processed by the compression engine 28, Which performs compression and decompression on the data."

Hovis, 3:8-12.

> "FIG. 4 shows a basic data flow structure of the memory 10 with a hardware assist compression engine 28. A memory control 24 interfaces With the memory 10, compression engine 28, and processor units 22. The purpose of a hardware based compression engine 28 is to provide the necessary bandwidth and latency required by memory entities that reside close to the processor data/instruction units (i.e. L1,L2,L3)."

Hovis, 3:41-46

> "The data is then uncompressed and sent back to the requesting element and uncompressed cache in the memory by the compression engine 28."

Hovis, 4:2-4.

> "If the cache is full, then the system must do a cast out and transfer the least recently used data element in the cache to the compressed storage (48), Which requires sending the data to the compression engine for compressing (46). The system then updates the data element in the cache (50) and continues as if the cache Was not full. If the cache Was not full, the system uses the setup table to retrieve the compressed data (34) and sends the compressed data to a compression engine for decompressing (38)."

Hovis, 4:34-44.

> "Aspects of the control of the hardware assisted memory compression can be implemented either by hardware, software (operating system, namely memory management), or a combination of both. The present invention is not dependent upon a particular hardware or software control scheme. The compression engine is preferably implemented in hardware in order to perform the compression and decompression in a timely fashion and With sufficient bandwidth."

Hovis, 5:4-11.

> "The implementation involves a hardware compression engine and the control can be via hardware, software, or a combination of both."

Hovis, 5:27-30.

51. The system of claim 6, wherein the first memory comprises: a physical memory.	Hovis, as evidenced by the example citations below, discloses "the system of claim 6, wherein the first memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the first memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 27 and 39 above.	

52. The system of claim 6, wherein the boot data the compressed form comprises: a plurality of files.	Hovis, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data the compressed form comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data the compressed form comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 16 and 23 above.	

53. The system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program.	Hovis, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 15, 17 and 24 above.	

59. The method of claim 8, wherein the operating system in the compressed form represents a plurality of files.	Hovis, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 16 and 23 above.	

60. The method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system.	Hovis, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 15, 17 and 24 above.	

Appendix C8
Invalidity of U.S. Patent 8,880,862 based on Hovis

63. The method of claim 8, wherein the second memory comprises: a physical memory.	Hovis, as evidenced by the example citations below, discloses "the method of claim 8, wherein the second memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the second memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 27 and 38 above.	

64. The method of claim 8, wherein the operating system comprises: a plurality of files.	Hovis, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 16 and 23 above.	

65. The method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program.	Hovis, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 15, 17 and 24 above.	

67. The method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access.	Hovis, as evidenced by the example citations below, discloses "the method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 31 above.	

71. The method of claim 11, wherein the boot data in the compressed form represents a plurality of files.	Hovis, as evidenced by the example citations below, discloses “the method of claim 11, wherein the boot data in the compressed form represents a plurality of files”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Hovis discloses this limitation: *See* Claims 16 and 23 above.	

72. The method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system.	Hovis, as evidenced by the example citations below, discloses "the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 15, 17 and 24 above.	

Appendix C8
Invalidity of U.S. Patent 8,880,862 based on Hovis

75. The method of claim 11, wherein the memory comprises: a physical memory.	Hovis, as evidenced by the example citations below, discloses "the method of claim 11, wherein the memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 27 above.	

76. The method of claim 11, wherein the operating system comprises: a plurality of files.	Hovis, as evidenced by the example citations below, discloses "the method of claim 11, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 16 and 23 above.	

77. The method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program.	Hovis, as evidenced by the example citations below, discloses "the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 15, 17 and 24 above.	

79. The method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access.	Hovis, as evidenced by the example citations below, discloses "the method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 31 above.	

80. The method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form.	Hovis, as evidenced by the example citations below, discloses "the method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 32, 33 and 49 above.	

81. The method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form.	Hovis, as evidenced by the example citations below, discloses "the method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 32, 33 and 49 above.	

83. The method of claim 13, wherein the boot data in the compressed form represents a plurality of files.	Hovis, as evidenced by the example citations below, discloses "the method of claim 13, wherein the boot data in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 16 and 23 above.	

84. The method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system.	Hovis, as evidenced by the example citations below, discloses "the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 15, 17 and 24 above.	

87. The method of claim 13, wherein the memory comprises: a physical memory.	Hovis, as evidenced by the example citations below, discloses "the method of claim 13, wherein the memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 27 above.	

Appendix C8
Invalidity of U.S. Patent 8,880,862 based on Hovis

88. The method of claim 13, wherein the operating system comprises: a plurality of files.	Hovis, as evidenced by the example citations below, discloses "the method of claim 13, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 16 and 23 above.	

89. The method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program.	Hovis, as evidenced by the example citations below, discloses "the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 15, 17 and 24 above.	

91. The method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access.	Hovis, as evidenced by the example citations below, discloses "the method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 31 above.	

92. The method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data.	Hovis, as evidenced by the example citations below, discloses "the method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 32, 33 and 49 above.	

93. The method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data.	Hovis, as evidenced by the example citations below, discloses "the method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claim 32, 33, and 49 above.	

Appendix C8
Invalidity of U.S. Patent 8,880,862 based on Hovis

97.1 The method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list,	Hovis, as evidenced by the example citations below, discloses "the method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 9.1-9.3 and 19 above.	

97.2 and wherein the updating comprises: associating the additional compressed boot data with the boot data list.	Hovis, as evidenced by the example citations below, discloses "and wherein the updating comprises: associating the additional compressed boot data with the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, and wherein the updating comprises: associating the additional compressed boot data with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 1.4.1, 2, 4, 5.6, and 8.6 above.	

98. The method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list.	Hovis, as evidenced by the example citations below, discloses "the method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 1.4.1, 5.6, and 8.6 above. "Also, the directory for the uncompressed cache must be updated when new locations are added or when old (the least recently used) locations are removed to compressed memory 16." Hovis, 4:18-21.	

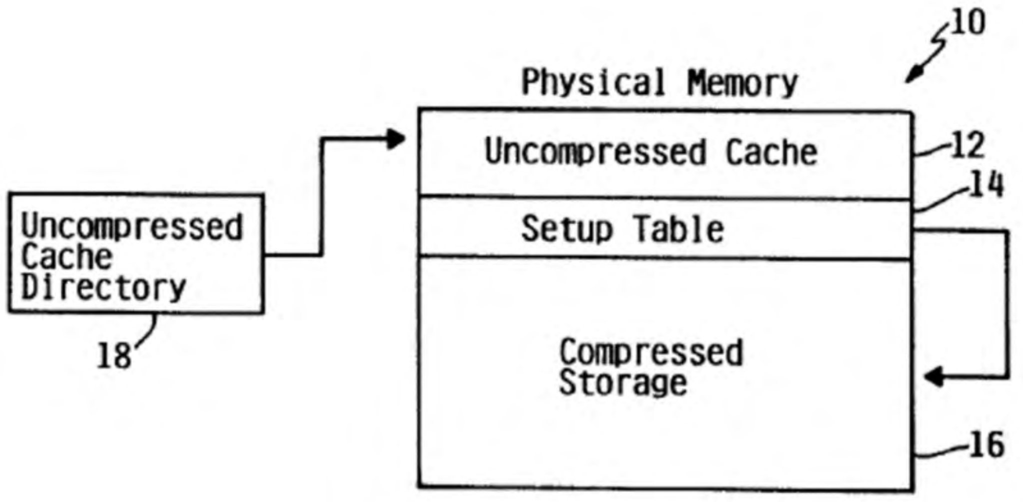
107. The method of claim 2, further comprising: storing the updated boot list in a non-volatile memory.	Hovis, as evidenced by the example citations below, discloses "the method of claim 2, further comprising: storing the updated boot list in a non-volatile memory."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 2, further comprising: storing the updated boot list in a non-volatile memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Hovis discloses this limitation:

See Claims 1.4.1, 5.6, and 8.6 above.

See also **Add disclosure where storing the updated boot list in a non-volatile memory.**



Hovis, Fig. 1

> "The setup table is used for specifying locations of compressed data stored within the compressed storage."

Hovis, Abstract. *See also* Hovis, 1:44-45.

108. The method of claim 2, further comprising: compressing at least a portion of the additional boot data.	Hovis, as evidenced by the example citations below, discloses "the method of claim 2, further comprising: compressing at least a portion of the additional boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 2, further comprising: compressing at least a portion of the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 1.1, 5, 9.1 and 8 above.	

109. The method of claim 108, further comprising: storing the compressed additional boot data.	Hovis, as evidenced by the example citations below, discloses "the method of claim 108, further comprising: storing the compressed additional boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 108, further comprising: storing the compressed additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Hovis discloses this limitation: *See* Claims 5.1, 8.1 and 9.1 above.	

Appendix C9
Invalidity of U.S. Patent 8,880,862 based on Ingvar

G.B. Patent No. 2276257 to Ingvar ("Ingvar") invalidates claims 1-6, 8-11, 13-17, 19, 23-24, 27-29, 31-33, 35-36, 39-40, 43-45, 47-53, 59-60, 63-65, 67, 71-72, 75-77, 79-81, 83-84, 87-89, 91-93, 97-98, and 107-109 of United States Patent No. 8,880,862 ("the '862 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '862 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix C9
Invalidity of U.S. Patent 8,880,862 based on Ingvar

1 (Preamble) A method for providing accelerated loading of an operating system in a computer system, the method comprising:	Ingvar, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, the method comprising:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system in a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: "A method for enabling the transfer of stored firmware during start-up of a computer, comprising the steps of reading configuration information stored in a air configuration table in the first memory device (40), transferring selected software modules from the first memory device to a second memory device (130, 60, 50), and storing the selected software modules at selected address areas in the second memory device; in all accordance with the configuration information." Ingvar, Abstract. "In order to provide quicker access to the program modules, these modules are normally copied from the PROM-packages to a system-contained main memory which has shorter access times." Ingvar at 6.	

Appendix C9
Invalidity of U.S. Patent 8,880,862 based on Ingvar

1.1 loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory;	Ingvar, as evidenced by the example citations below, discloses "loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ingvar discloses this limitation:

> "A method for enabling the transfer of stored firmware during start-up of a computer, comprising the steps of reading configuration information stored in a air configuration table in the first memory device (40), transferring selected software modules from the first memory device to a second memory device (130, 60, 50), and storing the selected software modules at selected address areas in the second memory device; in all accordance with the configuration information."

Ingvar, Abstract.

> "The memory device 40 may include firmware, such as startup software and BIOS program modules. According to the invention, certain software modules may be stored in a compressed form in the memory device 40. A compressed data packet which includes one or more software modules will take up less memory space than that taken up by a corresponding amount of data or information stored in an uncompressed state, as is well known to the person skilled in this art."

Ingvar at 10.

> "Steps S220 and S230 The selected software is compressed in Step S220 and is stored in Step S230 in a selected memory area in the first memory device 40, which will retain its data content in the absence of an applied voltage."

Ingvar at 18.

> "According to this inventive method, selected modules are compressed

> before making this compilation so that the data packet which is later loaded into the permanent memory device will have precisely the content that one wished to store in the permanent memory device 40."

Ingvar at 19.

Appendix C9
Invalidity of U.S. Patent 8,880,862 based on Ingvar

1.2 accessing the loaded portion of the boot data in the compressed form from memory;	Ingvar, as evidenced by the example citations below, discloses "accessing the loaded portion of the boot data in the compressed form from memory;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded portion of the boot data in the compressed form from memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ingvar discloses this limitation:

> "Step 5140 In Step S140, the first startup program summons a decompression or decompaction program which unpacks or decompresses the firmware modules that are given in the configuration table and that are stored in a compressed state. According to another embodiment of the inventive method, the first decompression program decompresses all firmware modules stored in the memory device 40, wherein the choice of those modules that shall be used is made in accordance with the configuration table after said decompression."

Ingvar at 12.

> "Step S240 In Step S240, there is inserted a decompression program, which can also be stored in the first memory device 40."

Ingvar at 18.

1.3 decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form; and	Ingvar, as evidenced by the example citations below, discloses "decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ingvar discloses this limitation:

> "Step 5140 In Step S140, the first startup program summons a decompression or decompaction program which unpacks or decompresses the firmware modules that are given in the configuration table and that are stored in a compressed state. According to another embodiment of the inventive method, the first decompression program decompresses all firmware modules stored in the memory device 40, wherein the choice of those modules that shall be used is made in accordance with the configuration table after said decompression."

Ingvar at 12.

> "Step S240 In Step S240, there is inserted a decompression program, which can also be stored in the first memory device 40."

Ingvar at 18.

1.4.1 updating the boot data list,	Ingvar, as evidenced by the example citations below, discloses "updating the boot data list,"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ingvar discloses this limitation:

> "A further object of the present invention is to provide a method to enable program modules or data tables which are stored as firmware when starting-up the computer system to be repositioned such that the remaining unoccupied memory space can be readily utilized."

Ingvar at 7.

1.4.2 wherein the decompressed portion of boot data comprises a portion of the operating system.	Ingvar, as evidenced by the example citations below, discloses “wherein the decompressed portion of boot data comprises a portion of the operating system.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the decompressed portion of boot data comprises a portion of the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Ingvar discloses this limitation: “Step 5140 In Step S140, the first startup program summons a decompression or decompaction program which unpacks or decompresses the firmware modules that are given in the configuration table and that are stored in a compressed state. According to another embodiment of the inventive method, the first decompression program decompresses all firmware modules stored in the memory device 40, wherein the choice of those modules that shall be used is made in accordance with the configuration table after said decompression.” Ingvar at 12. “Step S240 In Step S240, there is inserted a decompression program, which can also be stored in the first memory device 40.” Ingvar at 18.	

Appendix C9
Invalidity of U.S. Patent 8,880,862 based on Ingvar

2. The method of claim 1, wherein the updating comprises: associating additional boot data with the boot data list.	Ingvar, as evidenced by the example citations below, discloses "wherein the updating comprises: associating additional boot data with the boot data list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein associating additional boot data with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1.4.1 above.	

3. The method of claim 1, wherein the updating comprises: removing an association of additional boot data that is associated with the boot data list from the boot data list.	Ingvar, as evidenced by the example citations below, discloses "wherein the updating comprises: removing an association of additional boot data that is associated with the boot data list from the boot data list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein removing an association of additional boot data that is associated with the boot data list from the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1.4.1 above.	

4. The method of claim 1, wherein the updating comprises: associating additional boot data with the boot data list; and compressing a portion of the additional boot data.	Ingvar, as evidenced by the example citations below, discloses "wherein the updating comprises: associating additional boot data with the boot data list; and compressing a portion of the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein associating additional boot data with the boot data list; and compressing a portion of the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 1.4.1, and 2 above.	

5 (Preamble) A method for booting a computer system, the method comprising:	Ingvar, as evidenced by the example citations below, discloses "a method for booting a computer system, the method comprising:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ingvar discloses this limitation:

See Claims 1-1.4.2 above.

See also **Add disclosure from '608 Patent Claims 1.1 and 1.2.**

> "A method for enabling the transfer of stored firmware during start-up of a computer, comprising the steps of reading configuration information stored in a air configuration table in the first memory device (40), transferring selected software modules from the first memory device to a second memory device (130, 60, 50), and storing the selected software modules at selected address areas in the second memory device; in all accordance with the configuration information."

Ingvar, Abstract.

> "The resultant configuration data or information can be stored in a configuration table which will thus disclose those program modules which are required by the computer system in respect of the current configuration of said system."

Ingvar at 11-12. *See also* Ingvar at 12, ¶¶ 1-2.

> "The area c may include the aforesaid configuration table containing information as to which software modules shall be used, and information as to which addresses and software modules respectively shall be work-stored."

Ingvar at 13.

> "In Step S150, the startup program can use the information in the configuration table to decide which of the program modules shall be used in the computer system and to obtain information as to where, i.e. at which addresses, respective software modules shall be placed in the

> working memory 130 for future use, until the computer unit is stopped and restarted."

Ingvar at 14-15.

> "Step S250 According to one embodiment of the invention, there is inserted in Step S250 configuration information which may also be stored in the first memory device. The configuration information may have the form of a data table which discloses, among other things, the memory addresses at which the various software modules shall be stored when decompressed. The table may also include information as to which software modules shall be work-stored in the working memory and/or the addresses at which the modules shall be work-stored when carrying out the first method according to the invention, as described above."

Ingvar at 18.

> "According to one aspect on the present invention there is provided a method for use when starting up a computer system arrangement, the arrangement comprising at least one computer unit and at least one first memory device which includes at least one first data block with information stored as firmware, the method including the step of initiating the start of the computer unit and the step of transferring software modules from the first memory device to a second memory device, characterized by the steps of reading changeable configuration information stored in a configuration table in the first memory device; transferring selected modules among said software modules from the first memory device to the second memory device, and storing the selected software modules at selected address areas in the second memory device in accordance with the configuration information."

Ingvar at 10.

> "When starting-up the computer unit 20, the data processing device 30 reads information contained in a memory device 40."

Ingvar at 10.

> "Step S120 and S130 In Step S120, the data processing unit 30 included in the computer unit executes a first startup program. This startup program can be executed and when executed, the first startup program may include a routine which requires a check to be made as to which peripheral units are included in the hardware included in the computer system, as illustrated by Step S130 in Fig. 2."

Ingvar at 11.

Appendix C9
Invalidity of U.S. Patent 8,880,862 based on Ingvar

5.1 storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory;	Ingvar, as evidenced by the example citations below, discloses "storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1.1 above.	

5.2 loading the stored compressed boot data from the first memory;	Ingvar, as evidenced by the example citations below, discloses "loading the stored compressed boot data from the first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the stored compressed boot data from the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1.1 above.	

5.3 accessing the loaded compressed boot data;	Ingvar, as evidenced by the example citations below, discloses "accessing the loaded compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1.2 above.	

5.4 decompressing the accessed compressed boot data;	Ingvar, as evidenced by the example citations below, discloses "decompressing the accessed compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1.3 above.	

5.5 utilizing the decompressed boot data to at least partially boot the computer system;	Ingvar, as evidenced by the example citations below, discloses "utilizing the decompressed boot data to at least partially boot the computer system"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed boot data to at least partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ingvar discloses this limitation:

See Claims 1.3 and 1.4.2 above.

See also

> "Step 5140 In Step S140, the first startup program summons a decompression or decompaction program which unpacks or decompresses the firmware modules that are given in the configuration table and that are stored in a compressed state. According to another embodiment of the inventive method, the first decompression program decompresses all firmware modules stored in the memory device 40, wherein the choice of those modules that shall be used is made in accordance with the configuration table after said decompression."

Ingvar at 12.

> "Step S240 In Step S240, there is inserted a decompression program, which can also be stored in the first memory device 40."

Ingvar at 18.

5.6 updating the boot data list;	Ingvar, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1.4.1 above.	

5.7 wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form.	Ingvar, as evidenced by the example citations below, discloses "wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 1-1.3 above.	

6 (Preamble) A system comprising: a processor;	Ingvar, as evidenced by the example citations below, discloses “a system comprising: a processor:”

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system comprising: a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.

Ingvar discloses this limitation:

> “According to one aspect on the present invention there is provided a method for use when starting up a computer system arrangement, the arrangement comprising at least one computer unit and at least one first memory device which includes at least one first data block with information stored as firmware, the method including the step of initiating the start of the computer unit and the step of transferring software modules from the first memory device to a second memory device, characterized by the steps of reading changeable configuration information stored in a configuration table in the first memory device; transferring selected modules among said software modules from the first memory device to the second memory device, and storing the selected software modules at selected address areas in the second memory device in accordance with the configuration information.”

Ingvar at 10.

> “When starting-up the computer unit 20, the data processing device 30 reads information contained in a memory device 40.”

Ingvar at 10.

> “Step S120 and S130 In Step S120, the data processing unit 30 included in the computer unit executes a first startup program. This startup program can be executed and when executed, the first startup program may include a routine which requires a check to be made as to which peripheral units are included in the hardware included in the computer system, as illustrated by Step S130 in Fig. 2.”

Ingvar at 11.

6.1 a memory; and	Ingvar, as evidenced by the example citations below, discloses “a memory”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a memory.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 1.1 and 1.2 above.	

6.2 a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor	Ingvar, as evidenced by the example citations below, discloses "a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 1.1 and 1.2 above. "According to one aspect on the present invention there is provided a method for use when starting up a computer system arrangement, the arrangement comprising at least one computer unit and at least one first memory device which includes at least one first data block with information stored as firmware, the method including the step of initiating the start of the computer unit and the step of transferring software modules from the first memory device to a second memory device, characterized by the steps of reading changeable configuration information stored in a configuration table in the first memory device; transferring selected modules among said software modules from the first memory device to the second memory device, and storing the selected software modules at selected address areas in the second memory device in accordance with the configuration information." Ingvar at 10. "When starting-up the computer unit 20, the data processing device 30 reads information contained in a memory device 40." Ingvar at 10. "Step S120 and S130 In Step S120, the data processing unit 30 included in the computer unit executes a first startup program. This startup program can be executed and when executed, the first startup program may include a routine which requires a check to be made as to which peripheral units are included in the hardware included in the computer	

system, as illustrated by Step S130 in Fig. 2." Ingvar at 11.

6.3 wherein the processor is configured:	Ingvar, as evidenced by the example citations below, discloses "wherein the processor is configured:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the processor is configured), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ingvar discloses this limitation:

> "According to one aspect on the present invention there is provided a method for use when starting up a computer system arrangement, the arrangement comprising at least one computer unit and at least one first memory device which includes at least one first data block with information stored as firmware, the method including the step of initiating the start of the computer unit and the step of transferring software modules from the first memory device to a second memory device, characterized by the steps of reading changeable configuration information stored in a configuration table in the first memory device; transferring selected modules among said software modules from the first memory device to the second memory device, and storing the selected software modules at selected address areas in the second memory device in accordance with the configuration information."

Ingvar at 10.

> "When starting-up the computer unit 20, the data processing device 30 reads information contained in a memory device 40."

Ingvar at 10.

> "Step S120 and S130 In Step S120, the data processing unit 30 included in the computer unit executes a first startup program. This startup program can be executed and when executed, the first startup program may include a routine which requires a check to be made as to which peripheral units are included in the hardware included in the computer system, as illustrated by Step S130 in Fig. 2."

Ingvar at 11.

6.4 to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory,	Ingvar, as evidenced by the example citations below, discloses "to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1.1 above.	

6.5 to access the loaded portion of the boot data in the compressed form,	Ingvar, as evidenced by the example citations below, discloses "to access the loaded portion of the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to access the loaded portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1.2 above.	

6.6 to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data, and	Ingvar, as evidenced by the example citations below, discloses “to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1.3 above.	

6.7 to update the boot data list.	Ingvar, as evidenced by the example citations below, discloses "to update the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to update the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1.4.1 above.	

8 (Preamble) A method of loading an operating system for booting a computer system, comprising:	Ingvar, as evidenced by the example citations below, discloses "A method of loading an operating system for booting a computer system, comprising:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method of loading an operating system for booting a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1 (Preamble) above.	

8.1 storing a portion of the operating system in a compressed form in a first memory;	Ingvar, as evidenced by the example citations below, discloses "storing a portion of the operating system in a compressed form in a first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing a portion of the operating system in a compressed form in a first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 6.2 above.	

8.2 loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list;	Ingvar, as evidenced by the example citations below, discloses "loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1.1 above.	

8.3 accessing the loaded portion of the operating system from the second memory in the compressed form;	Ingvar, as evidenced by the example citations below, discloses "accessing the loaded portion of the operating system from the second memory in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded portion of the operating system from the second memory in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1.2 above.	

8.4 decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system;	Ingvar, as evidenced by the example citations below, discloses "decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 1.3 and 1.4.2 above.	

8.5 utilizing the decompressed portion of the operating system to at least partially boot the computer system; and	Ingvar, as evidenced by the example citations below, discloses "utilizing the decompressed portion of the operating system to at least partially boot the computer system"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed portion of the operating system to at least partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ingvar discloses this limitation:

See Claims 1.3 and 1.4.2 above.

See also

> "Step 5140 In Step S140, the first startup program summons a decompression or decompaction program which unpacks or decompresses the firmware modules that are given in the configuration table and that are stored in a compressed state. According to another embodiment of the inventive method, the first decompression program decompresses all firmware modules stored in the memory device 40, wherein the choice of those modules that shall be used is made in accordance with the configuration table after said decompression."

Ingvar at 12.

> "Step S240 In Step S240, there is inserted a decompression program, which can also be stored in the first memory device 40."

Ingvar at 18.

8.6 updating the boot data list,	Ingvar, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1.4.1 above.	

8.7 wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form.	Ingvar, as evidenced by the example citations below, discloses "wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 1-1.3 and 5.7 above.	

Appendix C9
Invalidity of U.S. Patent 8,880,862 based on Ingvar

9.1 The method of claim 8, further comprising: compressing an additional portion of the operating system that is not associated with the boot data list; and	Ingvar, as evidenced by the example citations below, discloses "the method of claim 8, further comprising: compressing an additional portion of the operating system that is not associated with the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, compressing an additional portion of the operating system that is not associated with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1.1 above.	

9.2 storing the additional portion of the operating system in the first memory, and	Ingvar, as evidenced by the example citations below, discloses “storing the additional portion of the operating system in the first memory”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing the additional portion of the operating system in the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 8.1 above.	

9.3 wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system.	Ingvar, as evidenced by the example citations below, discloses "wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 8.5 above.	

10. The method of claim 9, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 9, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ingvar discloses this limitation:

See Claim 9.1 above.

> "Yet another object is to reduce the amount of hardware required by a computer system, for instance such hardware as address decoders, jumpers, permanent memory sockets, etc."

Ingvar at 8.

> "The memory area c may also include the aforesaid decompression program, which can be used to decompress compressed data. The memory area d may be a memory area which contains compressed software modules."

Ingvar at 13.

> "According to this embodiment, when the configuration table is stored in a compressed state in the memory device 40, the reconfiguration command can provide access to the decompressed configuration table in the working memory 130. According to this latter embodiment of the invention, the user is able to initiate compression and storage of the modified table in the permanent memory device 40."

Ingvar at 17.

Appendix C9
Invalidity of U.S. Patent 8,880,862 based on Ingvar

11 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	Ingvar, as evidenced by the example citations below, discloses “a method for providing accelerated loading of an operating system in a computer system, comprising:”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1 (Preamble) above.	

11.1 loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system;	Ingvar, as evidenced by the example citations below, discloses "loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ingvar discloses this limitation:

See Claim 1.1 above.

See also

> "According to one aspect on the present invention there is provided a method for use when starting up a computer system arrangement, the arrangement comprising at least one computer unit and at least one first memory device which includes at least one first data block with information stored as firmware, the method including the step of initiating the start of the computer unit and the step of transferring software modules from the first memory device to a second memory device, characterized by the steps of reading changeable configuration information stored in a configuration table in the first memory device; transferring selected modules among said software modules from the first memory device to the second memory device, and storing the selected software modules at selected address areas in the second memory device in accordance with the configuration information."

Ingvar at 10.

> "When starting-up the computer unit 20, the data processing device 30 reads information contained in a memory device 40."

Ingvar at 10.

> "Step S120 and S130 In Step S120, the data processing unit 30 included in the computer unit executes a first startup program. This startup program can be executed and when executed, the first startup program

> may include a routine which requires a check to be made as to which peripheral units are included in the hardware included in the computer system, as illustrated by Step S130 in Fig. 2."

Ingvar at 11.

11.2 accessing the loaded boot data in compressed form from the memory;	Ingvar, as evidenced by the example citations below, discloses “accessing the loaded boot data in compressed form from the memory”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded boot data in compressed form from the memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1.2 above.	

Appendix C9
Invalidity of U.S. Patent 8,880,862 based on Ingvar

11.3 decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form;	Ingvar, as evidenced by the example citations below, discloses "decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1.3 above.	

11.4 utilizing the decompressed boot data to load at least a portion of the operating system for the computer system; and	Ingvar, as evidenced by the example citations below, discloses "utilizing the decompressed boot data to load at least a portion of the operating system for the computer system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed boot data to load at least a portion of the operating system for the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: "Step 5140 In Step S140, the first startup program summons a decompression or decompaction program which unpacks or decompresses the firmware modules that are given in the configuration table and that are stored in a compressed state. According to another embodiment of the inventive method, the first decompression program decompresses all firmware modules stored in the memory device 40, wherein the choice of those modules that shall be used is made in accordance with the configuration table after said decompression." Ingvar at 12. "Step S240 In Step S240, there is inserted a decompression program, which can also be stored in the first memory device 40." Ingvar at 18.	

11.5 updating the boot data list.	Ingvar, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1.4.1 above.	

Appendix C9
Invalidity of U.S. Patent 8,880,862 based on Ingvar

13 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	Ingvar, as evidenced by the example citations below, discloses “a method for providing accelerated loading of an operating system in a computer system, comprising”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1 (Preamble) above.	

Appendix C9
Invalidity of U.S. Patent 8,880,862 based on Ingvar

13.1 loading boot data in a compressed form that is associated with a boot data list from a boot device;	Ingvar, as evidenced by the example citations below, discloses "loading boot data in a compressed form that is associated with a boot data list from a boot device"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading boot data in a compressed form that is associated with a boot data list from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1.1 above.	

13.2 accessing the loaded boot data in the compressed form;	Ingvar, as evidenced by the example citations below, discloses "accessing the loaded boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1.2 above.	

13.3 decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form;	Ingvar, as evidenced by the example citations below, discloses "decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1.3 above.	

13.4 updating the boot data list.	Ingvar, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1.4.1 above.	

Appendix C9
Invalidity of U.S. Patent 8,880,862 based on Ingvar

14 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	Ingvar, as evidenced by the example citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, comprising"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1 (Preamble) above.	

14.1 accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list;	Ingvar, as evidenced by the example citations below, discloses "accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ingvar discloses this limitation:

See Claims 1.1 and 1.2 above.

See also

> "A method for enabling the transfer of stored firmware during start-up of a computer, comprising the steps of reading configuration information stored in a air configuration table in the first memory device (40), transferring selected software modules from the first memory device to a second memory device (130, 60, 50), and storing the selected software modules at selected address areas in the second memory device; in all accordance with the configuration information."

Ingvar, Abstract.

> "The resultant configuration data or information can be stored in a configuration table which will thus disclose those program modules which are required by the computer system in respect of the current configuration of said system."

Ingvar at 11-12. *See also* Ingvar at 12, ¶¶ 1-2.

> "The area c may include the aforesaid configuration table containing information as to which software modules shall be used, and information as to which addresses and software modules respectively shall be work-stored."

Ingvar at 13.

> "In Step S150, the startup program can use the information in the

> configuration table to decide which of the program modules shall be used in the computer system and to obtain information as to where, i.e. at which addresses, respective software modules shall be placed in the working memory 130 for future use, until the computer unit is stopped and restarted."

Ingvar at 14-15.

> "Step S250 According to one embodiment of the invention, there is inserted in Step S250 configuration information which may also be stored in the first memory device. The configuration information may have the form of a data table which discloses, among other things, the memory addresses at which the various software modules shall be stored when decompressed. The table may also include information as to which software modules shall be work-stored in the working memory and/or the addresses at which the modules shall be work-stored when carrying out the first method according to the invention, as described above."

Ingvar at 18.

14.2 loading the boot data into a memory; and	Ingvar, as evidenced by the example citations below, discloses "loading the boot data into a memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the boot data into a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1.1 above.	

14.3 servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form; and	Ingvar, as evidenced by the example citations below, discloses "servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Ingvar discloses this limitation:

See Claims 1.2 and 1.3 above.

See also

> "Step 5140 In Step S140, the first startup program summons a decompression or decompaction program which unpacks or decompresses the firmware modules that are given in the configuration table and that are stored in a compressed state. According to another embodiment of the inventive method, the first decompression program decompresses all firmware modules stored in the memory device 40, wherein the choice of those modules that shall be used is made in accordance with the configuration table after said decompression."

Ingvar at 12.

> "Step S240 In Step S240, there is inserted a decompression program, which can also be stored in the first memory device 40."

Ingvar at 18.

14.4 updating the boot data list.	Ingvar, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 1.4.1 above.	

15. The method of claim 14, wherein the boot data comprises: a program code associated with the operating system.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 14, wherein the boot data comprises: a program code associated with the operating system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the boot data comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 1 (Preamble) and 1.1 above.	

16. The method of claim 14, wherein the operating system comprises: a plurality of files.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 14, wherein the operating system comprises: a plurality of files."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

17. The method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 15 above.	

Appendix C9
Invalidity of U.S. Patent 8,880,862 based on Ingvar

19.1 The method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:	Ingvar, as evidenced by the example citations below, discloses "the method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

19.2 associating the accessed boot data that is not associated with the boot data list to the boot data list.	Ingvar, as evidenced by the example citations below, discloses "associating the accessed boot data that is not associated with the boot data list to the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, associating the accessed boot data that is not associated with the boot data list to the boot data list.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 2 above.	

23. The method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 16 above.	

24. The method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system.
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 15 above.	

27. The method of claim 1, wherein the memory comprises: a physical memory.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 1, wherein the memory comprises: a physical memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 1.1 and 1.2 above.	

28. The method of claim 1, wherein the operating system comprises: a plurality of files.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 1, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 16 above.	

29. The method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 15 and 17 above.	

31. The method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 1.1 and 1.2 above.	

32. The method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: "The memory area c may also include the aforesaid decompression program, which can be used to decompress compressed data. The memory area d may be a memory area which contains compressed software modules." Ingvar at 13.	

33. The method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form.	Ingvar, as evidenced by the example citations below, discloses “the method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 32 above.	

35. The method of claim 5, wherein the compressed boot data represents a plurality of files.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 5, wherein the compressed boot data represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the compressed boot data represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 16 and 23 above.	

Appendix C9
Invalidity of U.S. Patent 8,880,862 based on Ingvar

36. The method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 15 and 17 above.	

39. The method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 1.1 and 1.2 above.	

40. The method of claim 36, wherein the operating system comprises: a plurality of files.	Ingvar, as evidenced by the example citations below, discloses “the method of claim 36, wherein the operating system comprises: a plurality of files”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 36, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 16 and 23 above.	

43. The method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 31 above.	

44. The method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 32 above.	

45. The method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 32 and 33 above.	

47. The system of claim 6, wherein the boot data in the compressed form represents a plurality of files.	Ingvar, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 16 and 23 above.	

48. The system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system.	Ingvar, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 15, 17 and 24 above.	

49. The system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form.	Ingvar, as evidenced by the example citations below, discloses "the system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 10 above.	

50. The system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form.	Ingvar, as evidenced by the example citations below, discloses "the system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: "Yet another object is to reduce the amount of hardware required by a computer system, for instance such hardware as address decoders, jumpers, permanent memory sockets, etc." Ingvar at 8. "The memory area c may also include the aforesaid decompression program, which can be used to decompress compressed data. The memory area d may be a memory area which contains compressed software modules." Ingvar at 13. "According to this embodiment, when the configuration table is stored in a compressed state in the memory device 40, the reconfiguration command can provide access to the decompressed configuration table in the working memory 130. According to this latter embodiment of the invention, the user is able to initiate compression and storage of the modified table in the permanent memory device 40." Ingvar at 17.	

51. The system of claim 6, wherein the first memory comprises: a physical memory.	Ingvar, as evidenced by the example citations below, discloses "the system of claim 6, wherein the first memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the first memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 27 and 39 above.	

52. The system of claim 6, wherein the boot data the compressed form comprises: a plurality of files.	Ingvar, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data the compressed form comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data the compressed form comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 16 and 23 above.	

53. The system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program.	Ingvar, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 15, 17 and 24 above.	

59. The method of claim 8, wherein the operating system in the compressed form represents a plurality of files.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 16 and 23 above.	

60. The method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 15, 17 and 24 above.	

63. The method of claim 8, wherein the second memory comprises: a physical memory.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 8, wherein the second memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the second memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 27 and 38 above.	

64. The method of claim 8, wherein the operating system comprises: a plurality of files.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 16 and 23 above.	

65. The method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 15, 17 and 24 above.	

67. The method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 31 above.	

71. The method of claim 11, wherein the boot data in the compressed form represents a plurality of files.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 11, wherein the boot data in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 16 and 23 above.	

72. The method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 15, 17 and 24 above.	

75. The method of claim 11, wherein the memory comprises: a physical memory.	Ingvar, as evidenced by the example citations below, discloses “the method of claim 11, wherein the memory comprises: a physical memory”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 27 above.	

Appendix C9
Invalidity of U.S. Patent 8,880,862 based on Ingvar

76. The method of claim 11, wherein the operating system comprises: a plurality of files.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 11, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 16 and 23 above.	

77. The method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 15, 17 and 24 above.	

79. The method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 31 above.	

80. The method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 32, 33 and 49 above.	

81. The method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form.	Ingvar, as evidenced by the example citations below, discloses “the method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 32, 33 and 49 above.	

83. The method of claim 13, wherein the boot data in the compressed form represents a plurality of files.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 13, wherein the boot data in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 16 and 23 above.	

84. The method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 15, 17 and 24 above.	

87. The method of claim 13, wherein the memory comprises: a physical memory.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 13, wherein the memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 27 above.	

88. The method of claim 13, wherein the operating system comprises: a plurality of files.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 13, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 16 and 23 above.	

89. The method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 15, 17 and 24 above.	

91. The method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 31 above.	

92. The method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 32, 33 and 49 above.	

93. The method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claim 32, 33, and 49 above.	

97.1 The method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list,	Ingvar, as evidenced by the example citations below, discloses "the method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 9.1-9.3 and 19 above.	

97.2 and wherein the updating comprises: associating the additional compressed boot data with the boot data list.	Ingvar, as evidenced by the example citations below, discloses "and wherein the updating comprises: associating the additional compressed boot data with the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, and wherein the updating comprises: associating the additional compressed boot data with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 1.4.1, 2, 4, 5.6, and 8.6 above.	

98. The method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 1.4.1, 5.6, and 8.6 above.	

Appendix C9
Invalidity of U.S. Patent 8,880,862 based on Ingvar

107. The method of claim 2, further comprising: storing the updated boot list in a non-volatile memory.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 2, further comprising: storing the updated boot list in a non-volatile memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 2, further comprising: storing the updated boot list in a non-volatile memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 1.4.1, 5.6, and 8.6 above. :	

108. The method of claim 2, further comprising: compressing at least a portion of the additional boot data.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 2, further comprising: compressing at least a portion of the additional boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 2, further comprising: compressing at least a portion of the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 1.1, 5, 9.1 and 8 above.	

109. The method of claim 108, further comprising: storing the compressed additional boot data.	Ingvar, as evidenced by the example citations below, discloses "the method of claim 108, further comprising: storing the compressed additional boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 108, further comprising: storing the compressed additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Ingvar discloses this limitation: *See* Claims 5.1, 8.1 and 9.1 above.	

Appendix C10
Invalidity of U.S. Patent 8,880,862 based on Kikinis

PCT Application No. WO 94/19768 to Kikinis ("Kikinis") invalidates claims 1-6, 8-11, 13-17, 19, 23-24, 27-29, 31-33, 35-36, 39-40, 43-45, 47-53, 59-60, 63-65, 67, 71-72, 75-77, 79-81, 83-84, 87-89, 91-93, 97-98, and 107-109 of United States Patent No. 8,880,862 ("the '862 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '862 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix C10
Invalidity of U.S. Patent 8,880,862 based on Kikinis

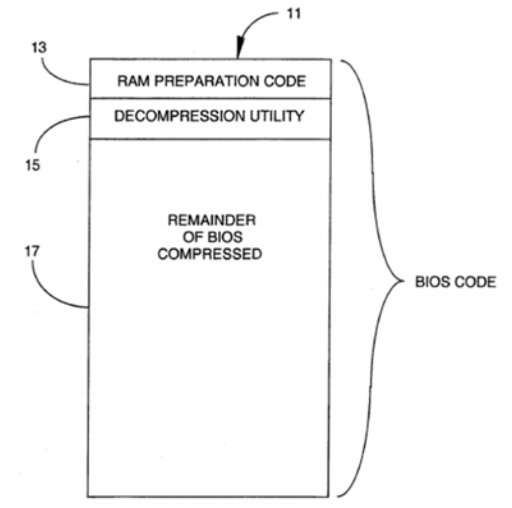
1 (Preamble) A method for providing accelerated loading of an operating system in a computer system, the method comprising:	Kikinis, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, the method comprising:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system in a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

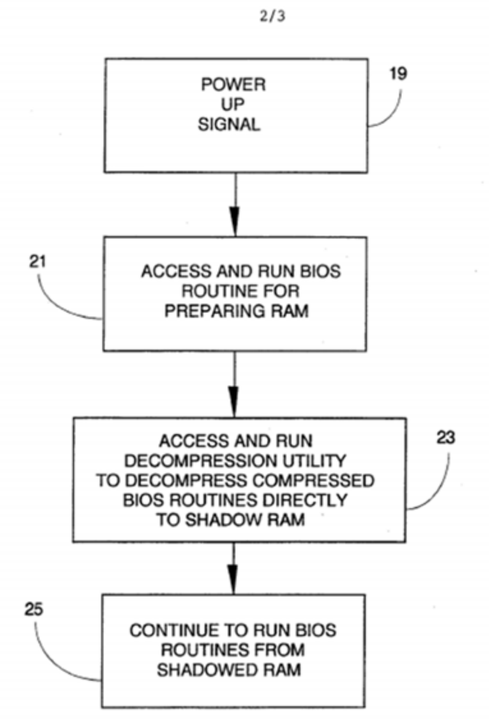
Kikinis discloses this claim limitation:

> A means of providing a BIOS routine from an EPROM to RAM in a general purpose computer, where the BIOS routine is greater in number of lines of code than the line capacity of the EPROM is provided. The BIOS routine is stored in EPROM in three portions (11). A first portion (13) is uncompressed, and loadable and operable by the CPU of the computer to initialize RAM. A second portion (17) is compressed by one of a number of schemes. A third portion (15) is a decompression utility loadable and operable by the computer CPU to decompress the compressed portion from EPROM and copy the resulting code to RAM, resulting in a BIOS in RAM (4).

Kikinis, Abstract



Kikinis, Fig. 1


2/3
POWER UP SIGNAL
19
21
ACCESS AND RUN BIOS ROUTINE FOR PREPARING RAM
ACCESS AND RUN DECOMPRESSION UTILITY TO DECOMPRESS COMPRESSED BIOS ROUTINES DIRECTLY TO SHADOW RAM
23
25
CONTINUE TO RUN BIOS ROUTINES FROM SHADOWED RAM

Kikinis, Fig 2

See also Kikinis Fig. 3.

1.1 loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory;	Kikinis, as evidenced by the example citations below, discloses "loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Kikinis discloses this claim limitation:

> A means of providing a BIOS routine from an EPROM to RAM in a general purpose computer, where the BIOS routine is greater in number of lines of code than the line capacity of the EPROM is provided. The BIOS routine is stored in EPROM in three portions (11). A first portion (13) is uncompressed, and loadable and operable by the CPU of the computer to initialize RAM. A second portion (17) is compressed by one of a number of schemes. A third portion (15) is a decompression utility loadable and operable by the computer CPU to decompress the compressed portion from EPROM and copy the resulting code to RAM, resulting in a BIOS in RAM (4).

Kikinis, Abstract

> As is well known in the art, a computer system, to be of any use, must comprise all the computer hardware, such as the CPU, memory devices, communication buses, and so forth. There must also be instruction sets (programs/software) for the CPU to follow to accomplish tasks. The programmed information (application software) is typically stored on an internal or peripheral memory device to be accessed by the CPU as needed.

Kikinis, 1

> A firmware device according to a preferred embodiment of the invention provides a BIOS routine for a general-purpose computer having a CPU microprocessor, comprising a programmable non-volatile memory device, and a BIOS routine stored on the programmable non-volatile memory device. The BIOS routine has a compressed portion, an uncompressed portion operable by the CPU to initialize random access

> memory in the general-purpose computer, and a decompression utility code operable by the CPU to load the compressed portion, decompress it, and copy the decompressed code to the random access memory. In a preferred embodiment the programmable memory device is an EPROM device

Kikinis, 2-3

> In most BIOS systems for general purpose computers, the BIOS is stored in an EPROM device as described in the background section above. Upon power up the BIOS initializes the system, doing basic tasks like accessing and checking the operation of on-board random-access memory RAM, and typically, somewhere during the initialization, at least a part of the BIOS code is copied (the BIOS copies itself) into a portion of the on-board RAM.

Kikinis, 4

> In typical general-purpose computers, as soon as the system receives power, the BIOS tests and initializes system RAM, then copies (shadows) itself from the EPROM to the RAM. The BIOS continues to run in RAM. The purpose of shadowing the BIOS in RAM is to give the CPU microprocessor much faster access to the BIOS code than it would have by accessing the EPROM every time a BIOS code sequence is needed in continuing operations.

Kikinis, 4

> The present invention comprises a means of compressing at least a significant portion of the BIOS code, storing all of the BIOS code, including the compressed portion, in EPROM, and releasing the compressed code on powerup, so all of the code is available for the computer to use. Also on powerup, the entire code is shadowed to RAM

Kikinis, 4-5

> Fig. 1 is a diagrammatical representation of a compressed BIOS 11 according to the present invention. There are three different portions of the code. Portion 13 is code to perform all operations to initialize and test the system RAM, and make it ready for use, and is a familiar portion of conventional BIOS routines. This portion in some applications needs to perform such functions as initializing and testing a memory controller and cache controllers and cache memory. Portion 15 is a decompression utility. Portion 17 represents the balance of the BIOS code in compressed form. It will be apparent to those with skill in

> the art that there are a number of compression schemes and related decompression routines that might be used.

Kikins 5

> Fig. 2 is a flow chart showing the operation of a computer from startup following a BIOS routine according to the present invention. From powerup signal 19, which is typically derived from the act of closing the power on switch, operation goes to initialization operation 21, during which system RAM is initialized. In operation 21, the system runs portion 13 of Fig. 1.
>
> Next, decompression utility 15 (Fig. 1) is accessed and run in operation 23. The decompression utility processes the balance of the BIOS code (compressed), translates it into operable code, and shadows it to system RAM. Although such decompression utilities are available, the code pointing to the compressed portion of the BIOS, and that which causes the decompressed code to be shadowed to RAM is not a part of a conventional decompression routine. These commands are added to the BIOS of the invention.
>
> After the BIOS is shadowed operation continues (25) from the BIOS in system RAM. All remaining BIOS processes, including testing and initializing the remainder of the computer subsystems are accomplished in this operating portion.

Kikinis, 5-6

> One means by which the BIOS code may be compressed is based on the fact that BIOS routines, as is common in most other coded instruction sets, make use of frequently repeated code sequences. EPROMs used for BIOS are typically byte-wide devices, that is, the device can store "words" of 8 bits. A sixteen bit word requires, then, two lines of BIOS code.

Kikinis, 6

See also Kikinis Fig. 1-3

1.2 accessing the loaded portion of the boot data in the compressed form from memory;	Kikinis, as evidenced by the example citations below, discloses "accessing the loaded portion of the boot data in the compressed form from memory;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded portion of the boot data in the compressed form from memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Kikinis discloses this claim limitation:

> A means of providing a BIOS routine from an EPROM to RAM in a general purpose computer, where the BIOS routine is greater in number of lines of code than the line capacity of the EPROM is provided. The BIOS routine is stored in EPROM in three portions (11). A first portion (13) is uncompressed, and loadable and operable by the CPU of the computer to initialize RAM. A second portion (17) is compressed by one of a number of schemes. A third portion (15) is a decompression utility loadable and operable by the computer CPU to decompress the compressed portion from EPROM and copy the resulting code to RAM, resulting in a BIOS in RAM (4).

Kikinis, Abstract

> As is well known in the art, a computer system, to be of any use, must comprise all the computer hardware, such as the CPU, memory devices, communication buses, and so forth. There must also be instruction sets (programs/software) for the CPU to follow to accomplish tasks. The programmed information (application software) is typically stored on an internal or peripheral memory device to be accessed by the CPU as needed.

Kikinis, 1

> A firmware device according to a preferred embodiment of the invention provides a BIOS routine for a general-purpose computer having a CPU microprocessor, comprising a programmable non-volatile memory device, and a BIOS routine stored on the programmable non-volatile memory device. The BIOS routine has a compressed portion, an uncompressed portion operable by the CPU to initialize random access memory in the general-purpose computer, and a decompression utility code operable by the CPU to load the compressed portion, decompress

> it, and copy the decompressed code to the random access memory. In a preferred embodiment the programmable memory device is an EPROM device

Kikinis, 2-3

> In most BIOS systems for general purpose computers, the BIOS is stored in an EPROM device as described in the background section above. Upon power up the BIOS initializes the system, doing basic tasks like accessing and checking the operation of on-board random-access memory RAM, and typically, somewhere during the initialization, at least a part of the BIOS code is copied (the BIOS copies itself) into a portion of the on-board RAM.

Kikinis, 4

> In typical general-purpose computers, as soon as the system receives power, the BIOS tests and initializes system RAM, then copies (shadows) itself from the EPROM to the RAM. The BIOS continues to run in RAM. The purpose of shadowing the BIOS in RAM is to give the CPU microprocessor much faster access to the BIOS code than it would have by accessing the EPROM every time a BIOS code sequence is needed in continuing operations.

Kikinis, 4

> The present invention comprises a means of compressing at least a significant portion of the BIOS code, storing all of the BIOS code, including the compressed portion, in EPROM, and releasing the compressed code on powerup, so all of the code is available for the computer to use. Also on powerup, the entire code is shadowed to RAM

Kikinis, 4-5

> Fig. 2 is a flow chart showing the operation of a computer from startup following a BIOS routine according to the present invention. From powerup signal 19, which is typically derived from the act of closing the power on switch, operation goes to initialization operation 21, during which system RAM is initialized. In operation 21, the system runs portion 13 of Fig. 1.
>
> Next, decompression utility 15 (Fig. 1) is accessed and run in operation 23. The decompression utility processes the balance of the BIOS code (compressed), translates it into operable code, and shadows it to system RAM. Although such decompression utilities are available, the code pointing to the compressed portion of the BIOS, and that which causes

> the decompressed code to be shadowed to RAM is not a part of a conventional decompression routine. These commands are added to the BIOS of the invention.
>
> After the BIOS is shadowed operation continues (25) from the BIOS in system RAM. All remaining BIOS processes, including testing and initializing the remainder of the computer subsystems are accomplished in this operating portion.

Kikinis, 5-6

> Fig. 3 is a diagrammatical representation of the token decompression scheme described above. After the BIOS according to the embodiment of the invention has initialized and tested the RAM, the decompression utility is booted, and begins to read the compressed portion of the BIOS at Start 27. At 29 the decompression utility loads the first/next byte from the compressed portion of
> the EPROM BIOS. If this byte is hex FF (31), it is recognized as a token, and control goes to 33,. where the system reads the byte following the token flag. This byte is always a pointer to a code sequence.

Kikinis, 7

See also Kikinis Fig. 1-3.

Appendix C10
Invalidity of U.S. Patent 8,880,862 based on Kikinis

1.3 decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form; and	Kikinis, as evidenced by the example citations below, discloses "decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Kikinis discloses this claim limitation:

> A means of providing a BIOS routine from an EPROM to RAM in a general purpose computer, where the BIOS routine is greater in number of lines of code than the line capacity of the EPROM is provided. The BIOS routine is stored in EPROM in three portions (11). A first portion (13) is uncompressed, and loadable and operable by the CPU of the computer to initialize RAM. A second portion (17) is compressed by one of a number of schemes. A third portion (15) is a decompression utility loadable and operable by the computer CPU to decompress the compressed portion from EPROM and copy the resulting code to RAM, resulting in a BIOS in RAM (4).

Kikinis, Abstract

> A firmware device according to a preferred embodiment of the invention provides a BIOS routine for a general-purpose computer having a CPU microprocessor, comprising a programmable non-volatile memory device, and a BIOS routine stored on the programmable non-volatile memory device. The BIOS routine has a compressed portion, an uncompressed portion operable by the CPU to initialize random access memory in the general-purpose computer, and a decompression utility code operable by the CPU to load the compressed portion, decompress it, and copy the decompressed code to the random access memory. In a preferred embodiment the programmable memory device is an EPROM device

Kikinis, 2-3

> In typical general-purpose computers, as soon as the system receives power, the BIOS tests and initializes system RAM, then copies (shadows) itself from the EPROM to the RAM. The BIOS continues to run in RAM. The purpose of shadowing the BIOS in RAM is to give the CPU microprocessor much faster access to the BIOS code than it would have by accessing the EPROM every time a BIOS code sequence is needed in continuing operations.

Kikinis, 4

> Fig. 2 is a flow chart showing the operation of a computer from startup following a BIOS routine according to the present invention. From powerup signal 19, which is typically derived from the act of closing the power on switch, operation goes to initialization operation 21, during which system RAM is initialized. In operation 21, the system runs portion 13 of Fig. 1.
>
> Next, decompression utility 15 (Fig. 1) is accessed and run in operation 23. The decompression utility processes the balance of the BIOS code (compressed), translates it into operable code, and shadows it to system RAM. Although such decompression utilities are available, the code pointing to the compressed portion of the BIOS, and that which causes the decompressed code to be shadowed to RAM is not a part of a conventional decompression routine. These commands are added to the BIOS of the invention.
>
> After the BIOS is shadowed operation continues (25) from the BIOS in system RAM. All remaining BIOS processes, including testing and initializing the remainder of the computer subsystems are accomplished in this operating portion.

Kikinis, 5-6

> Fig. 3 is a diagrammatical representation of the token decompression scheme described above. After the BIOS according to the embodiment of the invention has initialized and tested the RAM, the decompression utility is booted, and begins to read the compressed portion of the BIOS at Start 27. At 29 the decompression utility loads the first/next byte from the compressed portion of
> the EPROM BIOS. If this byte is hex FF (31), it is recognized as a token, and control goes to 33,. where the system reads the byte following the token flag. This byte is always a pointer to a code sequence.

Kikinis, 7 *See also* Kikinis Fig. 1-3.

1.4.1 updating the boot data list,	Kikinis, as evidenced by the example citations below, discloses "updating the boot data list,"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

1.4.2 wherein the decompressed portion of boot data comprises a portion of the operating system.	Kikinis, as evidenced by the example citations below, discloses "wherein the decompressed portion of boot data comprises a portion of the operating system."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the decompressed portion of boot data comprises a portion of the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Kikinis discloses this claim limitation:

> A means of providing a BIOS routine from an EPROM to RAM in a general purpose computer, where the BIOS routine is greater in number of lines of code than the line capacity of the EPROM is provided. The BIOS routine is stored in EPROM in three portions (11). A first portion (13) is uncompressed, and loadable and operable by the CPU of the computer to initialize RAM. A second portion (17) is compressed by one of a number of schemes. A third portion (15) is a decompression utility loadable and operable by the computer CPU to decompress the compressed portion from EPROM and copy the resulting code to RAM, resulting in a BIOS in RAM (4).

Kikinis, Abstract

> As is well known in the art, a computer system, to be of any use, must comprise all the computer hardware, such as the CPU, memory devices, communication buses, and so forth. There must also be instruction sets (programs/software) for the CPU to follow to accomplish tasks. The programmed information (application software) is typically stored on an internal or peripheral memory device to be accessed by the CPU as needed.

Kikinis, 1

> A firmware device according to a preferred embodiment of the invention provides a BIOS routine for a general-purpose computer having a CPU microprocessor, comprising a programmable non-volatile memory device, and a BIOS routine stored on the programmable non-volatile memory device. The BIOS routine has a compressed portion, an uncompressed portion operable by the CPU to initialize random access memory in the general-purpose computer, and a decompression utility code operable by the CPU to load the compressed portion, decompress

> it, and copy the decompressed code to the random access memory. In a preferred embodiment the programmable memory device is an EPROM device

Kikinis, 2-3

> In most BIOS systems for general purpose computers, the BIOS is stored in an EPROM device as described in the background section above. Upon power up the BIOS initializes the system, doing basic tasks like accessing and checking the operation of on-board random-access memory RAM, and typically, somewhere during the initialization, at least a part of the BIOS code is copied (the BIOS copies itself) into a portion of the on-board RAM.

Kikinis, 4

> In typical general-purpose computers, as soon as the system receives power, the BIOS tests and initializes system RAM, then copies (shadows) itself from the EPROM to the RAM. The BIOS continues to run in RAM. The purpose of shadowing the BIOS in RAM is to give the CPU microprocessor much faster access to the BIOS code than it would have by accessing the EPROM every time a BIOS code sequence is needed in continuing operations.

Kikinis, 4

> The present invention comprises a means of compressing at least a significant portion of the BIOS code, storing all of the BIOS code, including the compressed portion, in EPROM, and releasing the compressed code on powerup, so all of the code is available for the computer to use. Also on powerup, the entire code is shadowed to RAM

Kikinis, 4-5

Appendix C10
Invalidity of U.S. Patent 8,880,862 based on Kikinis

2. The method of claim 1, wherein the updating comprises: associating additional boot data with the boot data list.	Kikinis, as evidenced by the example citations below, discloses "wherein the updating comprises: associating additional boot data with the boot data list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein associating additional boot data with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

3. The method of claim 1, wherein the updating comprises: removing an association of additional boot data that is associated with the boot data list from the boot data list.	Kikinis, as evidenced by the example citations below, discloses “wherein the updating comprises: removing an association of additional boot data that is associated with the boot data list from the boot data list.”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein removing an association of additional boot data that is associated with the boot data list from the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.	

4. The method of claim 1, wherein the updating comprises: associating additional boot data with the boot data list; and compressing a portion of the additional boot data.	Kikinis, as evidenced by the example citations below, discloses "wherein the updating comprises: associating additional boot data with the boot data list; and compressing a portion of the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein associating additional boot data with the boot data list; and compressing a portion of the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

5 (Preamble) A method for booting a computer system, the method comprising:	Kikinis, as evidenced by the example citations below, discloses "a method for booting a computer system, the method comprising:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 1-1.4.2 above.	

5.1 storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory;	Kikinis, as evidenced by the example citations below, discloses "storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 1.1 above.	

5.2 loading the stored compressed boot data from the first memory;	Kikinis, as evidenced by the example citations below, discloses "loading the stored compressed boot data from the first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the stored compressed boot data from the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 1.1 above.	

5.3 accessing the loaded compressed boot data;	Kikinis, as evidenced by the example citations below, discloses "accessing the loaded compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 1.2 above.	

5.4 decompressing the accessed compressed boot data;	Kikinis, as evidenced by the example citations below, discloses "decompressing the accessed compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 1.3 above.	

5.5 utilizing the decompressed boot data to at least partially boot the computer system;	Kikinis, as evidenced by the example citations below, discloses "utilizing the decompressed boot data to at least partially boot the computer system"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed boot data to at least partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Kikinis discloses this limitation:

See Claims 1.3 and 1.4.2 above.

See also

Kikinis discloses this claim limitation:

> A means of providing a BIOS routine from an EPROM to RAM in a general purpose computer, where the BIOS routine is greater in number of lines of code than the line capacity of the EPROM is provided. The BIOS routine is stored in EPROM in three portions (11). A first portion (13) is uncompressed, and loadable and operable by the CPU of the computer to initialize RAM. A second portion (17) is compressed by one of a number of schemes. A third portion (15) is a decompression utility loadable and operable by the computer CPU to decompress the compressed portion from EPROM and copy the resulting code to RAM, resulting in a BIOS in RAM (4).

Kikinis, Abstract

> As is well known in the art, a computer system, to be of any use, must comprise all the computer hardware, such as the CPU, memory devices, communication buses, and so forth. There must also be instruction sets (programs/software) for the CPU to follow to accomplish tasks. The programmed information (application software) is typically stored on an internal or peripheral memory device to be accessed by the CPU as needed.

Kikinis, 1

> A firmware device according to a preferred embodiment of the invention provides a BIOS routine for a general-purpose computer having a CPU microprocessor, comprising a programmable non-volatile

> memory device, and a BIOS routine stored on the programmable non-volatile memory device. The BIOS routine has a compressed portion, an uncompressed portion operable by the CPU to initialize random access memory in the general-purpose computer, and a decompression utility code operable by the CPU to load the compressed portion, decompress it, and copy the decompressed code to the random access memory. In a preferred embodiment the programmable memory device is an EPROM device

Kikinis, 2-3

> In most BIOS systems for general purpose computers, the BIOS is stored in an EPROM device as described in the background section above. Upon power up the BIOS initializes the system, doing basic tasks like accessing and checking the operation of on-board random-access memory RAM, and typically, somewhere during the initialization, at least a part of the BIOS code is copied (the BIOS copies itself) into a portion of the on-board RAM.

Kikinis, 4

> In typical general-purpose computers, as soon as the system receives power, the BIOS tests and initializes system RAM, then copies (shadows) itself from the EPROM to the RAM. The BIOS continues to run in RAM. The purpose of shadowing the BIOS in RAM is to give the CPU microprocessor much faster access to the BIOS code than it would have by accessing the EPROM every time a BIOS code sequence is needed in continuing operations.

Kikinis, 4

> The present invention comprises a means of compressing at least a significant portion of the BIOS code, storing all of the BIOS code, including the compressed portion, in EPROM, and releasing the compressed code on powerup, so all of the code is available for the computer to use. Also on powerup, the entire code is shadowed to RAM

Kikinis, 4-5

> Fig. 2 is a flow chart showing the operation of a computer from startup following a BIOS routine according to the present invention. From powerup signal 19, which is typically derived from the act of closing the power on switch, operation goes to initialization operation 21, during which system RAM is initialized. In operation 21, the system runs portion 13 of Fig. 1.

> Next, decompression utility 15 (Fig. 1) is accessed and run in operation 23. The decompression utility processes the balance of the BIOS code (compressed), translates it into operable code, and shadows it to system RAM. Although such decompression utilities are available, the code pointing to the compressed portion of the BIOS, and that which causes the decompressed code to be shadowed to RAM is not a part of a conventional decompression routine. These commands are added to the BIOS of the invention.
>
> After the BIOS is shadowed operation continues (25) from the BIOS in system RAM. All remaining BIOS processes, including testing and initializing the remainder of the computer subsystems are accomplished in this operating portion.

Kikinis, 5-6

> Fig. 3 is a diagrammatical representation of the token decompression scheme described above. After the BIOS according to the embodiment of the invention has initialized and tested the RAM, the decompression utility is booted, and begins to read the compressed portion of the BIOS at Start 27. At 29 the decompression utility loads the first/next byte from the compressed portion of
> the EPROM BIOS. If this byte is hex FF (31), it is recognized as a token, and control goes to 33,. where the system reads the byte following the token flag. This byte is always a pointer to a code sequence.

Kikinis, 7

See also Kikinis Fig. 1-3.

5.6 updating the boot data list;	Kikinis, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

5.7 wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form.	Kikinis, as evidenced by the example citations below, discloses "wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 1-1.3 above.	

6 (Preamble) A system comprising: a processor;	Kikinis, as evidenced by the example citations below, discloses "a system comprising: a processor:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system comprising: a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Kikinis discloses this limitation:

> A means of providing a BIOS routine from an EPROM to RAM in a general purpose computer, where the BIOS routine is greater in number of lines of code than the line capacity of the EPROM is provided. The BIOS routine is stored in EPROM in three portions (11). A first portion (13) is uncompressed, and loadable and operable by the CPU of the computer to initialize RAM. A second portion (17) is compressed by one of a number of schemes. A third portion (15) is a decompression utility loadable and operable by the computer CPU to decompress the compressed portion from EPROM and copy the resulting code to RAM, resulting in a BIOS in RAM (4).

Kikinis, Abstract

> As is well known in the art, a computer system, to be of any use, must comprise all the computer hardware, such as the CPU, memory devices, communication buses, and so forth. There must also be instruction sets (programs/software) for the CPU to follow to accomplish tasks. The programmed information (application software) is typically stored on an internal or peripheral memory device to be accessed by the CPU as needed.

Kikinis, 1

> A firmware device according to a preferred embodiment of the invention provides a BIOS routine for a general-purpose computer having a CPU microprocessor, comprising a programmable non-volatile memory device, and a BIOS routine stored on the programmable non-volatile memory device. The BIOS routine has a compressed portion, an uncompressed portion operable by the CPU to initialize random access memory in the general-purpose computer, and a decompression utility code operable by the CPU to load the compressed portion, decompress it, and copy the decompressed code to the random access memory. In a

> preferred embodiment the programmable memory device is an EPROM device

Kikinis, 2-3

> In most BIOS systems for general purpose computers, the BIOS is stored in an EPROM device as described in the background section above. Upon power up the BIOS initializes the system, doing basic tasks like accessing and checking the operation of on-board random-access memory RAM, and typically, somewhere during the initialization, at least a part of the BIOS code is copied (the BIOS copies itself) into a portion of the on-board RAM.

Kikinis, 4

> In typical general-purpose computers, as soon as the system receives power, the BIOS tests and initializes system RAM, then copies (shadows) itself from the EPROM to the RAM. The BIOS continues to run in RAM. The purpose of shadowing the BIOS in RAM is to give the CPU microprocessor much faster access to the BIOS code than it would have by accessing the EPROM every time a BIOS code sequence is needed in continuing operations.

Kikinis, 4

> Fig. 1 is a diagrammatical representation of a compressed BIOS 11 according to the present invention. There are three different portions of the code. Portion 13 is code to perform all operations to initialize and test the system RAM, and make it ready for use, and is a familiar portion of conventional BIOS routines. This portion in some applications needs to perform such functions as initializing and testing a memory controller and cache controllers and cache memory. Portion 15 is a decompression utility. Portion 17 represents the balance of the BIOS code in compressed form. It will be apparent to those with skill in the art that there are a number of compression schemes and related decompression routines that might be used.

Kikins 5

> Fig. 2 is a flow chart showing the operation of a computer from startup following a BIOS routine according to the present invention. From powerup signal 19, which is typically derived from the act of closing the power on switch, operation goes to initialization operation 21, during which system RAM is initialized. In operation 21, the system runs portion 13 of Fig. 1.

> Next, decompression utility 15 (Fig. 1) is accessed and run in operation 23. The decompression utility processes the balance of the BIOS code (compressed), translates it into operable code, and shadows it to system RAM. Although such decompression utilities are available, the code pointing to the compressed portion of the BIOS, and that which causes the decompressed code to be shadowed to RAM is not a part of a conventional decompression routine. These commands are added to the BIOS of the invention.
>
> After the BIOS is shadowed operation continues (25) from the BIOS in system RAM. All remaining BIOS processes, including testing and initializing the remainder of the computer subsystems are accomplished in this operating portion.

Kikinis, 5-6

See also Kikinis Fig. 1-3.

6.1 a memory; and	Kikinis, as evidenced by the example citations below, discloses “a memory”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a memory.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 1.1 and 1.2 above.	

6.2 a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor	Kikinis, as evidenced by the example citations below, discloses “a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 1.1, 1.2, and Claim 6 (Preamble) above.	

Appendix C10
Invalidity of U.S. Patent 8,880,862 based on Kikinis

6.3 wherein the processor is configured:	Kikinis, as evidenced by the example citations below, discloses "wherein the processor is configured:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the processor is configured), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 6 (Preamble) above	

6.4 to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory,	Kikinis, as evidenced by the example citations below, discloses "to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 1.1 above.	

6.5 to access the loaded portion of the boot data in the compressed form,	Kikinis, as evidenced by the example citations below, discloses "to access the loaded portion of the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to access the loaded portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 1.2 above.	

6.6 to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data, and	Kikinis, as evidenced by the example citations below, discloses "to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 1.3 above.	

6.7 to update the boot data list.	Kikinis, as evidenced by the example citations below, discloses “to update the boot data list”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to update the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.	

Appendix C10
Invalidity of U.S. Patent 8,880,862 based on Kikinis

8 (Preamble) A method of loading an operating system for booting a computer system, comprising:	Kikinis, as evidenced by the example citations below, discloses "A method of loading an operating system for booting a computer system, comprising:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method of loading an operating system for booting a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 1 (Preamble) above.	

Appendix C10
Invalidity of U.S. Patent 8,880,862 based on Kikinis

8.1 storing a portion of the operating system in a compressed form in a first memory;	Kikinis, as evidenced by the example citations below, discloses “storing a portion of the operating system in a compressed form in a first memory”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing a portion of the operating system in a compressed form in a first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 6.2 above.	

8.2 loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list;	Kikinis, as evidenced by the example citations below, discloses “loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 1.1 above.	

8.3 accessing the loaded portion of the operating system from the second memory in the compressed form;	Kikinis, as evidenced by the example citations below, discloses “accessing the loaded portion of the operating system from the second memory in the compressed form”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded portion of the operating system from the second memory in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 1.2 above.	

8.4 decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system;	Kikinis, as evidenced by the example citations below, discloses "decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 1.3 and 1.4.2 above.	

8.5 utilizing the decompressed portion of the operating system to at least partially boot the computer system; and	Kikinis, as evidenced by the example citations below, discloses "utilizing the decompressed portion of the operating system to at least partially boot the computer system"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed portion of the operating system to at least partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Kikinis discloses this limitation:

See Claims 1.3 and 1.4.2 above.

See also

> A means of providing a BIOS routine from an EPROM to RAM in a general purpose computer, where the BIOS routine is greater in number of lines of code than the line capacity of the EPROM is provided. The BIOS routine is stored in EPROM in three portions (11). A first portion (13) is uncompressed, and loadable and operable by the CPU of the computer to initialize RAM. A second portion (17) is compressed by one of a number of schemes. A third portion (15) is a decompression utility loadable and operable by the computer CPU to decompress the compressed portion from EPROM and copy the resulting code to RAM, resulting in a BIOS in RAM (4).

Kikinis, Abstract

> As is well known in the art, a computer system, to be of any use, must comprise all the computer hardware, such as the CPU, memory devices, communication buses, and so forth. There must also be instruction sets (programs/software) for the CPU to follow to accomplish tasks. The programmed information (application software) is typically stored on an internal or peripheral memory device to be accessed by the CPU as needed.

Kikinis, 1

> A firmware device according to a preferred embodiment of the invention provides a BIOS routine for a general-purpose computer having a CPU microprocessor, comprising a programmable non-volatile

> memory device, and a BIOS routine stored on the programmable non-volatile memory device. The BIOS routine has a compressed portion, an uncompressed portion operable by the CPU to initialize random access memory in the general-purpose computer, and a decompression utility code operable by the CPU to load the compressed portion, decompress it, and copy the decompressed code to the random access memory. In a preferred embodiment the programmable memory device is an EPROM device

Kikinis, 2-3

> In most BIOS systems for general purpose computers, the BIOS is stored in an EPROM device as described in the background section above. Upon power up the BIOS initializes the system, doing basic tasks like accessing and checking the operation of on-board random-access memory RAM, and typically, somewhere during the initialization, at least a part of the BIOS code is copied (the BIOS copies itself) into a portion of the on-board RAM.

Kikinis, 4

> In typical general-purpose computers, as soon as the system receives power, the BIOS tests and initializes system RAM, then copies (shadows) itself from the EPROM to the RAM. The BIOS continues to run in RAM. The purpose of shadowing the BIOS in RAM is to give the CPU microprocessor much faster access to the BIOS code than it would have by accessing the EPROM every time a BIOS code sequence is needed in continuing operations.

Kikinis, 4

> Fig. 1 is a diagrammatical representation of a compressed BIOS 11 according to the present invention. There are three different portions of the code. Portion 13 is code to perform all operations to initialize and test the system RAM, and make it ready for use, and is a familiar portion of conventional BIOS routines. This portion in some applications needs to perform such functions as initializing and testing a memory controller and cache controllers and cache memory. Portion 15 is a decompression utility. Portion 17 represents the balance of the BIOS code in compressed form. It will be apparent to those with skill in the art that there are a number of compression schemes and related decompression routines that might be used.

Kikins 5

> Fig. 2 is a flow chart showing the operation of a computer from startup following a BIOS routine according to the present invention. From powerup signal 19, which is typically derived from the act of closing the power on switch, operation goes to initialization operation 21, during which system RAM is initialized. In operation 21, the system runs portion 13 of Fig. 1.
>
> Next, decompression utility 15 (Fig. 1) is accessed and run in operation 23. The decompression utility processes the balance of the BIOS code (compressed), translates it into operable code, and shadows it to system RAM. Although such decompression utilities are available, the code pointing to the compressed portion of the BIOS, and that which causes the decompressed code to be shadowed to RAM is not a part of a conventional decompression routine. These commands are added to the BIOS of the invention.
>
> After the BIOS is shadowed operation continues (25) from the BIOS in system RAM. All remaining BIOS processes, including testing and initializing the remainder of the computer subsystems are accomplished in this operating portion.

Kikinis, 5-6

See also Kikinis Fig. 1-3.

8.6 updating the boot data list,	Kikinis, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

8.7 wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form.	Kikinis, as evidenced by the example citations below, discloses "wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 1-1.3 and 5.7 above.	

9.1 The method of claim 8, further comprising: compressing an additional portion of the operating system that is not associated with the boot data list; and	Kikinis, as evidenced by the example citations below, discloses "the method of claim 8, further comprising: compressing an additional portion of the operating system that is not associated with the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, compressing an additional portion of the operating system that is not associated with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 1.1 above.	

9.2 storing the additional portion of the operating system in the first memory, and	Kikinis, as evidenced by the example citations below, discloses "storing the additional portion of the operating system in the first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing the additional portion of the operating system in the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 8.1 above.	

9.3 wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system.	Kikinis, as evidenced by the example citations below, discloses "wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 8.5 above.	

10. The method of claim 9, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 9, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 9.1 above.	

Appendix C10
Invalidity of U.S. Patent 8,880,862 based on Kikinis

11 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	Kikinis, as evidenced by the example citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, comprising:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 1 (Preamble) above.	

11.1 loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system;	Kikinis, as evidenced by the example citations below, discloses "loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 1.1 and 6 (Preamble) above.	

11.2 accessing the loaded boot data in compressed form from the memory;	Kikinis, as evidenced by the example citations below, discloses "accessing the loaded boot data in compressed form from the memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded boot data in compressed form from the memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 1.2 above.	

11.3 decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form;	Kikinis, as evidenced by the example citations below, discloses "decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 1.3 above.	

11.4 utilizing the decompressed boot data to load at least a portion of the operating system for the computer system; and	Kikinis, as evidenced by the example citations below, discloses "utilizing the decompressed boot data to load at least a portion of the operating system for the computer system"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed boot data to load at least a portion of the operating system for the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Kikinis discloses this limitation:

> A means of providing a BIOS routine from an EPROM to RAM in a general purpose computer, where the BIOS routine is greater in number of lines of code than the line capacity of the EPROM is provided. The BIOS routine is stored in EPROM in three portions (11). A first portion (13) is uncompressed, and loadable and operable by the CPU of the computer to initialize RAM. A second portion (17) is compressed by one of a number of schemes. A third portion (15) is a decompression utility loadable and operable by the computer CPU to decompress the compressed portion from EPROM and copy the resulting code to RAM, resulting in a BIOS in RAM (4).

Kikinis, Abstract

> As is well known in the art, a computer system, to be of any use, must comprise all the computer hardware, such as the CPU, memory devices, communication buses, and so forth. There must also be instruction sets (programs/software) for the CPU to follow to accomplish tasks. The programmed information (application software) is typically stored on an internal or peripheral memory device to be accessed by the CPU as needed.

Kikinis, 1

> A firmware device according to a preferred embodiment of the invention provides a BIOS routine for a general-purpose computer having a CPU microprocessor, comprising a programmable non-volatile memory device, and a BIOS routine stored on the programmable non-volatile memory device. The BIOS routine has a compressed portion, an uncompressed portion operable by the CPU to initialize random access memory in the general-purpose computer, and a decompression utility

code operable by the CPU to load the compressed portion, decompress it, and copy the decompressed code to the random access memory. In a preferred embodiment the programmable memory device is an EPROM device

Kikinis, 2-3

In most BIOS systems for general purpose computers, the BIOS is stored in an EPROM device as described in the background section above. Upon power up the BIOS initializes the system, doing basic tasks like accessing and checking the operation of on-board random-access memory RAM, and typically, somewhere during the initialization, at least a part of the BIOS code is copied (the BIOS copies itself) into a portion of the on-board RAM.

Kikinis, 4

In typical general-purpose computers, as soon as the system receives power, the BIOS tests and initializes system RAM, then copies (shadows) itself from the EPROM to the RAM. The BIOS continues to run in RAM. The purpose of shadowing the BIOS in RAM is to give the CPU microprocessor much faster access to the BIOS code than it would have by accessing the EPROM every time a BIOS code sequence is needed in continuing operations.

Kikinis, 4

Fig. 1 is a diagrammatical representation of a compressed BIOS 11 according to the present invention. There are three different portions of the code. Portion 13 is code to perform all operations to initialize and test the system RAM, and make it ready for use, and is a familiar portion of conventional BIOS routines. This portion in some applications needs to perform such functions as initializing and testing a memory controller and cache controllers and cache memory. Portion 15 is a decompression utility. Portion 17 represents the balance of the BIOS code in compressed form. It will be apparent to those with skill in the art that there are a number of compression schemes and related decompression routines that might be used.

Kikins 5

Fig. 2 is a flow chart showing the operation of a computer from startup following a BIOS routine according to the present invention. From powerup signal 19, which is typically derived from the act of closing the power on switch, operation goes to initialization operation 21, during which system RAM is initialized. In operation 21, the system runs portion 13 of Fig. 1.

Next, decompression utility 15 (Fig. 1) is accessed and run in operation 23. The decompression utility processes the balance of the BIOS code (compressed), translates it into operable code, and shadows it to system RAM. Although such decompression utilities are available, the code pointing to the compressed portion of the BIOS, and that which causes the decompressed code to be shadowed to RAM is not a part of a conventional decompression routine. These commands are added to the BIOS of the invention.

After the BIOS is shadowed operation continues (25) from the BIOS in system RAM. All remaining BIOS processes, including testing and initializing the remainder of the computer subsystems are accomplished in this operating portion.

Kikinis, 5-6

See also Kikinis Fig. 1-3.

11.5 updating the boot data list.	Kikinis, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

13 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	Kikinis, as evidenced by the example citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, comprising"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 1 (Preamble) above.	

13.1 loading boot data in a compressed form that is associated with a boot data list from a boot device;	Kikinis, as evidenced by the example citations below, discloses “loading boot data in a compressed form that is associated with a boot data list from a boot device”

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading boot data in a compressed form that is associated with a boot data list from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.

Kikinis discloses this limitation:

See Claim 1.1 above.

13.2 accessing the loaded boot data in the compressed form;	Kikinis, as evidenced by the example citations below, discloses "accessing the loaded boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 1.2 above.	

13.3 decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form;	Kikinis, as evidenced by the example citations below, discloses "decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 1.3 above.	

13.4 updating the boot data list.	Kikinis, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

14 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	Kikinis, as evidenced by the example citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, comprising"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 1 (Preamble) above.	

14.1 accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list;	Kikinis, as evidenced by the example citations below, discloses "accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Kikinis discloses this limitation:

See Claims 1.1 and 1.2 above.

See also

> A means of providing a BIOS routine from an EPROM to RAM in a general purpose computer, where the BIOS routine is greater in number of lines of code than the line capacity of the EPROM is provided. The BIOS routine is stored in EPROM in three portions (11). A first portion (13) is uncompressed, and loadable and operable by the CPU of the computer to initialize RAM. A second portion (17) is compressed by one of a number of schemes. A third portion (15) is a decompression utility loadable and operable by the computer CPU to decompress the compressed portion from EPROM and copy the resulting code to RAM, resulting in a BIOS in RAM (4).

Kikinis, Abstract

> A firmware device according to a preferred embodiment of the invention provides a BIOS routine for a general-purpose computer having a CPU microprocessor, comprising a programmable non-volatile memory device, and a BIOS routine stored on the programmable non-volatile memory device. The BIOS routine has a compressed portion, an uncompressed portion operable by the CPU to initialize random access memory in the general-purpose computer, and a decompression utility code operable by the CPU to load the compressed portion, decompress it, and copy the decompressed code to the random access memory. In a preferred embodiment the programmable memory device is an EPROM device

Kikinis, 2-3

> In typical general-purpose computers, as soon as the system receives power, the BIOS tests and initializes system RAM, then copies (shadows) itself from the EPROM to the RAM. The BIOS continues to run in RAM. The purpose of shadowing the BIOS in RAM is to give the CPU microprocessor much faster access to the BIOS code than it would have by accessing the EPROM every time a BIOS code sequence is needed in continuing operations.

Kikinis, 4

> Fig. 2 is a flow chart showing the operation of a computer from startup following a BIOS routine according to the present invention. From powerup signal 19, which is typically derived from the act of closing the power on switch, operation goes to initialization operation 21, during which system RAM is initialized. In operation 21, the system runs portion 13 of Fig. 1.
>
> Next, decompression utility 15 (Fig. 1) is accessed and run in operation 23. The decompression utility processes the balance of the BIOS code (compressed), translates it into operable code, and shadows it to system RAM. Although such decompression utilities are available, the code pointing to the compressed portion of the BIOS, and that which causes the decompressed code to be shadowed to RAM is not a part of a conventional decompression routine. These commands are added to the BIOS of the invention.
>
> After the BIOS is shadowed operation continues (25) from the BIOS in system RAM. All remaining BIOS processes, including testing and initializing the remainder of the computer subsystems are accomplished in this operating portion.

Kikinis, 5-6

See also Kikinis Fig. 1-3.

14.2 loading the boot data into a memory; and	Kikinis, as evidenced by the example citations below, discloses "loading the boot data into a memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the boot data into a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 1.1 above.	

14.3 servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form; and	Kikinis, as evidenced by the example citations below, discloses "servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Kikinis discloses this limitation:

See Claims 1.2 and 1.3 above.

See also

> A means of providing a BIOS routine from an EPROM to RAM in a general purpose computer, where the BIOS routine is greater in number of lines of code than the line capacity of the EPROM is provided. The BIOS routine is stored in EPROM in three portions (11). A first portion (13) is uncompressed, and loadable and operable by the CPU of the computer to initialize RAM. A second portion (17) is compressed by one of a number of schemes. A third portion (15) is a decompression utility loadable and operable by the computer CPU to decompress the compressed portion from EPROM and copy the resulting code to RAM, resulting in a BIOS in RAM (4).

Kikinis, Abstract

> As is well known in the art, a computer system, to be of any use, must comprise all the computer hardware, such as the CPU, memory devices, communication buses, and so forth. There must also be instruction sets (programs/software) for the CPU to follow to accomplish tasks. The programmed information (application software) is typically stored on an

internal or peripheral memory device to be accessed by the CPU as needed.

Kikinis, 1

A firmware device according to a preferred embodiment of the invention provides a BIOS routine for a general-purpose computer having a CPU microprocessor, comprising a programmable non-volatile memory device, and a BIOS routine stored on the programmable non-volatile memory device. The BIOS routine has a compressed portion, an uncompressed portion operable by the CPU to initialize random access memory in the general-purpose computer, and a decompression utility code operable by the CPU to load the compressed portion, decompress it, and copy the decompressed code to the random access memory. In a preferred embodiment the programmable memory device is an EPROM device

Kikinis, 2-3

In most BIOS systems for general purpose computers, the BIOS is stored in an EPROM device as described in the background section above. Upon power up the BIOS initializes the system, doing basic tasks like accessing and checking the operation of on-board random-access memory RAM, and typically, somewhere during the initialization, at least a part of the BIOS code is copied (the BIOS copies itself) into a portion of the on-board RAM.

Kikinis, 4

In typical general-purpose computers, as soon as the system receives power, the BIOS tests and initializes system RAM, then copies (shadows) itself from the EPROM to the RAM. The BIOS continues to run in RAM. The purpose of shadowing the BIOS in RAM is to give the CPU microprocessor much faster access to the BIOS code than it would have by accessing the EPROM every time a BIOS code sequence is needed in continuing operations.

Kikinis, 4

The present invention comprises a means of compressing at least a significant portion of the BIOS code, storing all of the BIOS code, including the compressed portion, in EPROM, and releasing the compressed code on powerup, so all of the code is available for the computer to use. Also on powerup, the entire code is shadowed to RAM

Kikinis, 4-5

> Fig. 2 is a flow chart showing the operation of a computer from startup following a BIOS routine according to the present invention. From powerup signal 19, which is typically derived from the act of closing the power on switch, operation goes to initialization operation 21, during which system RAM is initialized. In operation 21, the system runs portion 13 of Fig. 1.
>
> Next, decompression utility 15 (Fig. 1) is accessed and run in operation 23. The decompression utility processes the balance of the BIOS code (compressed), translates it into operable code, and shadows it to system RAM. Although such decompression utilities are available, the code pointing to the compressed portion of the BIOS, and that which causes the decompressed code to be shadowed to RAM is not a part of a conventional decompression routine. These commands are added to the BIOS of the invention.
>
> After the BIOS is shadowed operation continues (25) from the BIOS in system RAM. All remaining BIOS processes, including testing and initializing the remainder of the computer subsystems are accomplished in this operating portion.

Kikinis, 5-6

> Fig. 3 is a diagrammatical representation of the token decompression scheme described above. After the BIOS according to the embodiment of the invention has initialized and tested the RAM, the decompression utility is booted, and begins to read the compressed portion of the BIOS at Start 27. At 29 the decompression utility loads the first/next byte from the compressed portion of
> the EPROM BIOS. If this byte is hex FF (31), it is recognized as a token, and control goes to 33,. where the system reads the byte following the token flag. This byte is always a pointer to a code sequence.

Kikinis, 7

See also Kikinis Fig. 1-3.

14.4 updating the boot data list.	Kikinis, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

15. The method of claim 14, wherein the boot data comprises: a program code associated with the operating system.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 14, wherein the boot data comprises: a program code associated with the operating system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the boot data comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 1 (Preamble) and 1.1 above.	

16. The method of claim 14, wherein the operating system comprises: a plurality of files.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 14, wherein the operating system comprises: a plurality of files."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 1 and 14 above.	

17. The method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 15 above.	

19.1 The method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:	Kikinis, as evidenced by the example citations below, discloses "the method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 1.1 and 1.2 above	

19.2 associating the accessed boot data that is not associated with the boot data list to the boot data list.	Kikinis, as evidenced by the example citations below, discloses "associating the accessed boot data that is not associated with the boot data list to the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, associating the accessed boot data that is not associated with the boot data list to the boot data list.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 1 and 2 above.	

23. The method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 16 above.	

24. The method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system.
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 15 above.	

27. The method of claim 1, wherein the memory comprises: a physical memory.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 1, wherein the memory comprises: a physical memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 1.1 and 1.2 above	

28. The method of claim 1, wherein the operating system comprises: a plurality of files.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 1, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 16 above.	

29. The method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 15 and 17 above.	

31. The method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Kikinis discloses this limitation:

See Claims 1.1 and 1.2 above

See also

> Most BIOS implementations in general purpose computers are implemented in single chip erasable programmable non-volatile (EPROM) memory devices resident on a "motherboard" in the computer. There are other suitable non-volatile memory devices, however, which have been or may be used, including but not limited to EEPROM devices, "Flash Card" memories known in the art, masked ROM devices, CMOS RAM with a battery backup, and magnetic bubble memory.

Kikinis, 2

> A firmware device according to a preferred embodiment of the invention provides a BIOS routine for α general-purpose computer having a CPU microprocessor, comprising a programmable non-volatiJe memory device, and a BIOS routine stored on the programmable non-volatile memory device. The BIOS routine has a compressed portion, an uncompressed portion operable by the CPU to initialize random access memory in the general-purpose computer, and a decompression utility code operable by the CPU to load the compressed portion, decompress it, and copy the decompressed code to the random access memory. In a preferred embodiment the programmable memory device is an EPROM device.

Kikinis, 2-3

> For example, there are many non-volatile memories in which a compressed BIOS may be stored, retrieved, and decompressed, and several of these have been listed above. The fact of compressing the routines in the BIOS, including a loadable decompression routine, extends the capacity of any such finite non-volatile memory devices and hence the size of a BIOS routine that may be stored thereon.

Kikinis, 7-8

32. The method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 1.1 above *See also* > In a particular compression scheme compression is accomplished by a substituting a two line token for a longer sequence, where the longer sequence is a sequence often repeated in the BIOS. The value of the first line is a flag to the decompression routine that the next line is to be used to correlate to the longer sequence and copy that longer sequence in decompression. Kikinis, 3 > It is also true that there are a truly large number of compression schemes that might be employed to compress a portion of the BIOS. The invention should not be limited by the specific code relationship determined to compress the BIOS code. Kikinis, 8	

33. The method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Kikinis discloses this limitation:

See Claims 1.1 and 32 above.

See also

> In a particular compression scheme compression is accomplished by a substituting a two line token for a longer sequence, where the longer sequence is a sequence often repeated in the BIOS. The value of the first line is a flag to the decompression routine that the next line is to be used to correlate to the longer sequence and copy that longer sequence in decompression.

Kikinis, 3

> It is also true that there are a truly large number of compression schemes that might be employed to compress a portion of the BIOS. The invention should not be limited by the specific code relationship determined to compress the BIOS code.

Kikinis, 8

35. The method of claim 5, wherein the compressed boot data represents a plurality of files.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 5, wherein the compressed boot data represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the compressed boot data represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 16 and 23 above.	

36. The method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 15 and 17 above.	

39. The method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory.	Kikinis, as evidenced by the example citations below, discloses “the method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 1.1 and 1.2 above	

40. The method of claim 36, wherein the operating system comprises: a plurality of files.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 36, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 36, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 16 and 23 above.	

43. The method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access.	Kikinis, as evidenced by the example citations below, discloses “the method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 31 above.	

44. The method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 32 above.	

45. The method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 32 and 33 above.	

47. The system of claim 6, wherein the boot data in the compressed form represents a plurality of files.	Kikinis, as evidenced by the example citations below, discloses “the system of claim 6, wherein the boot data in the compressed form represents a plurality of files”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 16 and 23 above.	

48. The system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system.	Kikinis, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 15, 17 and 24 above.	

49. The system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form.	Kikinis, as evidenced by the example citations below, discloses "the system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 10 above.	

50. The system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form.	Kikinis, as evidenced by the example citations below, discloses "the system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Kikinis discloses this limitation:

> A means of providing a BIOS routine from an EPROM to RAM in a general purpose computer, where the BIOS routine is greater in number of lines of code than the line capacity of the EPROM is provided. The BIOS routine is stored in EPROM in three portions (11). A first portion (13) is uncompressed, and loadable and operable by the CPU of the computer to initialize RAM. A second portion (17) is compressed by one of a number of schemes. A third portion (15) is a decompression utility loadable and operable by the computer CPU to decompress the compressed portion from EPROM and copy the resulting code to RAM, resulting in a BIOS in RAM (4).

Kikinis, Abstract

> As is well known in the art, a computer system, to be of any use, must comprise all the computer hardware, such as the CPU, memory devices, communication buses, and so forth. There must also be instruction sets (programs/software) for the CPU to follow to accomplish tasks. The programmed information (application software) is typically stored on an internal or peripheral memory device to be accessed by the CPU as needed.

Kikinis, 1

> A firmware device according to a preferred embodiment of the invention provides a BIOS routine for a general-purpose computer having a CPU microprocessor, comprising a programmable non-volatile memory device, and a BIOS routine stored on the programmable non-volatile memory device. The BIOS routine has a compressed portion, an uncompressed portion operable by the CPU to initialize random access

> memory in the general-purpose computer, and a decompression utility code operable by the CPU to load the compressed portion, decompress it, and copy the decompressed code to the random access memory. In a preferred embodiment the programmable memory device is an EPROM device

Kikinis, 2-3

> In most BIOS systems for general purpose computers, the BIOS is stored in an EPROM device as described in the background section above. Upon power up the BIOS initializes the system, doing basic tasks like accessing and checking the operation of on-board random-access memory RAM, and typically, somewhere during the initialization, at least a part of the BIOS code is copied (the BIOS copies itself) into a portion of the on-board RAM.

Kikinis, 4

> In typical general-purpose computers, as soon as the system receives power, the BIOS tests and initializes system RAM, then copies (shadows) itself from the EPROM to the RAM. The BIOS continues to run in RAM. The purpose of shadowing the BIOS in RAM is to give the CPU microprocessor much faster access to the BIOS code than it would have by accessing the EPROM every time a BIOS code sequence is needed in continuing operations.

Kikinis, 4

> The present invention comprises a means of compressing at least a significant portion of the BIOS code, storing all of the BIOS code, including the compressed portion, in EPROM, and releasing the compressed code on powerup, so all of the code is available for the computer to use. Also on powerup, the entire code is shadowed to RAM

Kikinis, 4-5

> Fig. 2 is a flow chart showing the operation of a computer from startup following a BIOS routine according to the present invention. From powerup signal 19, which is typically derived from the act of closing the power on switch, operation goes to initialization operation 21, during which system RAM is initialized. In operation 21, the system runs portion 13 of Fig. 1.
>
> Next, decompression utility 15 (Fig. 1) is accessed and run in operation 23. The decompression utility processes the balance of the BIOS code

> (compressed), translates it into operable code, and shadows it to system RAM. Although such decompression utilities are available, the code pointing to the compressed portion of the BIOS, and that which causes the decompressed code to be shadowed to RAM is not a part of a conventional decompression routine. These commands are added to the BIOS of the invention.
>
> After the BIOS is shadowed operation continues (25) from the BIOS in system RAM. All remaining BIOS processes, including testing and initializing the remainder of the computer subsystems are accomplished in this operating portion.

Kikinis, 5-6

> Fig. 3 is a diagrammatical representation of the token decompression scheme described above. After the BIOS according to the embodiment of the invention has initialized and tested the RAM, the decompression utility is booted, and begins to read the compressed portion of the BIOS at Start 27. At 29 the decompression utility loads the first/next byte from the compressed portion of
> the EPROM BIOS. If this byte is hex FF (31), it is recognized as a token, and control goes to 33,. where the system reads the byte following the token flag. This byte is always a pointer to a code sequence.

Kikinis, 7

See also Kikinis Fig. 1-3.

51. The system of claim 6, wherein the first memory comprises: a physical memory.	Kikinis, as evidenced by the example citations below, discloses "the system of claim 6, wherein the first memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the first memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 27 and 39 above.	

52. The system of claim 6, wherein the boot data the compressed form comprises: a plurality of files.	Kikinis, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data the compressed form comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data the compressed form comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 16 and 23 above.	

53. The system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program.	Kikinis, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 15, 17 and 24 above.	

59. The method of claim 8, wherein the operating system in the compressed form represents a plurality of files.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 16 and 23 above.	

60. The method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system.	Kikinis, as evidenced by the example citations below, discloses “the method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 15, 17 and 24 above.	

63. The method of claim 8, wherein the second memory comprises: a physical memory.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 8, wherein the second memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the second memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 27 and 38 above.	

64. The method of claim 8, wherein the operating system comprises: a plurality of files.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 16 and 23 above.	

65. The method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 15, 17 and 24 above.	

67. The method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 31 above.	

Appendix C10
Invalidity of U.S. Patent 8,880,862 based on Kikinis

71. The method of claim 11, wherein the boot data in the compressed form represents a plurality of files.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 11, wherein the boot data in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 16 and 23 above.	

72. The method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 15, 17 and 24 above.	

75. The method of claim 11, wherein the memory comprises: a physical memory.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 11, wherein the memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 27 above.	

76. The method of claim 11, wherein the operating system comprises: a plurality of files.	Kikinis, as evidenced by the example citations below, discloses “the method of claim 11, wherein the operating system comprises: a plurality of files”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 16 and 23 above.	

77. The method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 15, 17 and 24 above.	

79. The method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 31 above.	

80. The method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 32, 33 and 49 above.	

81. The method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 32, 33 and 49 above.	

Appendix C10
Invalidity of U.S. Patent 8,880,862 based on Kikinis

83. The method of claim 13, wherein the boot data in the compressed form represents a plurality of files.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 13, wherein the boot data in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 16 and 23 above.	

84. The method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 15, 17 and 24 above.	

87. The method of claim 13, wherein the memory comprises: a physical memory.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 13, wherein the memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 27 above.	

88. The method of claim 13, wherein the operating system comprises: a plurality of files.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 13, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 16 and 23 above.	

Appendix C10
Invalidity of U.S. Patent 8,880,862 based on Kikinis

89. The method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 15, 17 and 24 above.	

91. The method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 31 above.	

92. The method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 32, 33 and 49 above.	

93. The method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claim 32, 33, and 49 above.	

97.1 The method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list,	Kikinis, as evidenced by the example citations below, discloses "the method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 9.1-9.3 and 19 above.	

97.2 and wherein the updating comprises: associating the additional compressed boot data with the boot data list.	Kikinis, as evidenced by the example citations below, discloses "and wherein the updating comprises: associating the additional compressed boot data with the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, and wherein the updating comprises: associating the additional compressed boot data with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

Appendix C10
Invalidity of U.S. Patent 8,880,862 based on Kikinis

98. The method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

107. The method of claim 2, further comprising: storing the updated boot list in a non-volatile memory.	Kikinis, as evidenced by the example citations below, discloses “the method of claim 2, further comprising: storing the updated boot list in a non-volatile memory.”

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 2, further comprising: storing the updated boot list in a non-volatile memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.

Kikinis discloses this limitation:

See Claims 1.4.1, 5.6, and 8.6 above.

See also

> Most BIOS implementations in general purpose computers are implemented in single chip erasable programmable non-volatile (EPROM) memory devices resident on a "motherboard" in the computer. There are other suitable non-volatile memory devices, however, which have been or may be used, including but not limited to EEPROM devices, "Flash Card" memories known in the art, masked ROM devices, CMOS RAM with a battery backup, and magnetic bubble memory.

Kikinis, 2

> A firmware device according to a preferred embodiment of the invention provides a BIOS routine for α general-purpose computer having a CPU microprocessor, comprising a programmable non-volatiJe memory device, and a BIOS routine stored on the programmable non-volatile memory device. The BIOS routine has a compressed portion, an uncompressed portion operable by the CPU to initialize random access memory in the general-purpose computer, and a decompression utility code operable by the CPU to load the compressed portion, decompress it, and copy the decompressed code to the random access memory. In a preferred embodiment the programmable memory device is an EPROM device.

Kikinis, 2-3

> For example, there are many non-volatile memories in which a compressed BIOS may be stored, retrieved, and decompressed, and several of these have been listed above. The fact of compressing the routines in the BIOS, including a loadable decompression routine, extends the capacity of any such finite non-volatile memory devices and hence the size of a BIOS routine that may be stored thereon.

Kikinis, 7-8

108. The method of claim 2, further comprising: compressing at least a portion of the additional boot data.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 2, further comprising: compressing at least a portion of the additional boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 2, further comprising: compressing at least a portion of the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Kikinis discloses this limitation: *See* Claims 1.1, 5, 9.1 and 8 above.	

109. The method of claim 108, further comprising: storing the compressed additional boot data.	Kikinis, as evidenced by the example citations below, discloses "the method of claim 108, further comprising: storing the compressed additional boot data"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 108, further comprising: storing the compressed additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Kikinis discloses this limitation:

See Claims 5.1, 8.1 and 9.1 above.

Appendix C11
Invalidity of U.S. Patent 8,880,862 based on Krocker

U.S. Patent No. 6,073,232 to Krocker ("Krocker") invalidates claims 1-6, 8-11, 13-17, 19, 23-24, 27-29, 31-33, 35-36, 39-40, 43-45, 47-53, 59-60, 63-65, 67, 71-72, 75-77, 79-81, 83-84, 87-89, 91-93, 97-98, and 107-109 of United States Patent No. 8,880,862 ("the '862 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

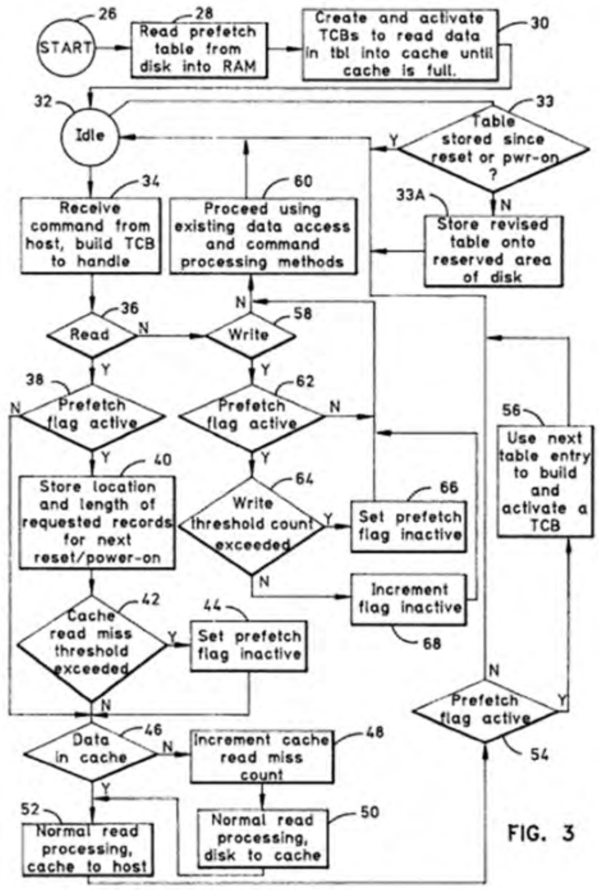
The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '862 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix C11
Invalidity of U.S. Patent 8,880,862 based on Krocker

1 (Preamble) A method for providing accelerated loading of an operating system in a computer system, the method comprising:	KROCKER, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, the method comprising:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system in a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

KROCKER discloses this limitation:



START 26
Read prefetch table from disk into RAM 28
Create and activate TCBs to read data in tbl into cache until cache is full. 30
Idle 32
Table stored since reset or pwr-on ? 33
Y
N
Store revised table onto reserved area of disk 33A
Receive command from host, build TCB to handle 34
Proceed using existing data access and command processing methods 60
Read 36
Write 58
Prefetch flag active 38
Prefetch flag active 62
Store location and length of requested records for next reset/power-on 40
Write threshold count exceeded 64
Set prefetch flag inactive 66
Increment flag inactive 68
Cache read miss threshold exceeded 42
Set prefetch flag inactive 44
Use next table entry to build and activate a TCB 56
Prefetch flag active 54
Data in cache 46
Increment cache read miss count 48
Normal read processing, cache to host 52
Normal read processing, disk to cache 50

FIG. 3

Kroeker, Fig. 3.

> "A method for increasing boot speed of a host computer with associated hard disk drive generates a prefetch table that contains pointers to disk locations and lengths of the records of an application program requested

> by the host computer during an initial power-on/reset."

Kroeker, Abstract.

> "Accordingly, it is an object of the present invention to provide a method for rapidly communicating a computer program from a disk drive to a host computer."

Kroeker, 2:66-67.

1.1 loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory;	KROCKER, as evidenced by the example citations below, discloses "loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

KROCKER discloses this limitation:

> "During the next power-on/reset, before the host computer is ready for data but after the disk drive has completed its reset routine, using the prefetch table the disk drive accesses the previously requested data and copies it onto the cache of the disk drive, from where it is transferred to the host computer when the host computer requests it."

Kroeker, Abstract.

> "This invention is realized in a critical machine component that causes a digital processing apparatus to adaptively store a computer program on the cache of the hard disk drive and communicate the program to the host computer."

Kroeker, 2:23-26.

> "Then, after a subsequent power-on or reset of the hard disk drive, and during a second power-on or reset of the host computer, the prefetch table is accessed to read into the data cache the data records."

Kroeker, 2:37-40.

> "In still another aspect, a computer hard disk drive includes at least one data storage disk and a data storage cache. Furthermore, the hard disk drive includes means for recording onto the cache, immediately after a hardware reset of the hard disk drive, data on the disk that has been requested by a host computer during a first hardware reset of the host computer."

Kroeker, 3:30-36.

> "From block 44, or from decision diamonds 38 or 42 when the decisions there are negative, the process moves to decision diamond 46 to determine whether the requested data exists in cache."

Kroeker, 5:65-6:1.

> "From block 50, or from decision diamond 46 if it was determined that the requested data exists in cache, the process moves to block 52 to transfer the record from cache 18 to the host computer 14."

Kroeker, 5:65-6:1.

1.2 accessing the loaded portion of the boot data in the compressed form from memory;	KROCKER, as evidenced by the example citations below, discloses "accessing the loaded portion of the boot data in the compressed form from memory;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded portion of the boot data in the compressed form from memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

KROCKER discloses this limitation:

> "In response to a subsequent read command from the host computer, it is determined whether records requested by the subsequent read command are stored in the data cache. If they are, the records are communicated from the cache to the host computer; otherwise, the records are communicated from the disk to the host computer."

Kroeker, 2:40-46.

> "Preferably, the accessing and determining steps are repeated for each power-on or reset of the host computer."

Kroeker, 2:47-48.

> "Additionally, the disk drive includes means for communicating the data from the cache to the host computer during a second hardware reset of the host computer."

Kroeker, 3:36-38.

> "If it is active, the logic, at block 56, uses the next entry in the prefetch table to build a task control block (TCB) to fetch data into the same segment of the cache 18 that the just-transferred record had occupied prior to being communicated to the host computer 14. In accordance with the present invention, the TCB in block 50 is activated as though a command otherwise was received across the device/file interface. In other words, when the host computer 14 is a PC, the TCB in block 50 is activated as though a command otherwise was received across the SCSI (or IDE)--disk drive interface. In this way, the relatively small amount of cache storage space can be optimally used during the adaptive caching process until all records designated in the prefetch table have been loaded

> into cache and then transferred to the host computer 14."

Kroeker, 6:16-31.

> "If the command is not a write command, the process moves to block 60 to proceed using existing data access and command processing methods, and then the process continues back to the idle state 32."

Kroeker, 6:44-48.

1.3 decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form; and	KROCKER, as evidenced by the example citations below, discloses "decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form; and"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

1.4.1 updating the boot data list,	KROCKER, as evidenced by the example citations below, discloses "updating the boot data list,"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

KROCKER discloses this limitation:

> "The prefetch table is updated to reflect disk location changes for the various records, or to reflect new records that were requested by the host computer but not found in cache during the previous power-on/reset."

Kroeker, Abstract.

> "In accordance with the present invention, steps executed by the digital processing apparatus include, after an initial power-up or reset of the hard disk drive and a host computer associated with the drive, receiving an initial read command from the host computer for transferring to the host computer a plurality of data records of a program stored on the disk. A prefetch table is then generated, with the table representing a disk location and length of each data record requested by the initial read command."

Kroeker, 2:20-37.

> "Additionally, the steps further include updating the data prefetch table, communicating the records from the disk to the host computer when the read counter exceeds a predetermined threshold, and setting a prefetch flag to inactive when the read counter exceeds the predetermined threshold."

Kroeker, 2:59-64.

> "When the command is a read command, code means generate a prefetch table representative of at least the disk location of the records requested by the read command for transfer of the records from the disk to the cache for a subsequent power-on or reset of the host computer. Moreover, code means are provided for determining whether the records have been stored in the cache in response to a previous power-on or reset of the host computer, and the records are communicated to the host computer in response."

Kroeker, 3:21-29.

> "As discussed further below, the prefetch table contains a listing of the disk locations and lengths of data records that were requested by the host computer 14 in the immediately previous power-on/reset. Additionally, a copy of a prefetch flag, if enabled by the user, is created and set active at block 28."

Kroeker, 3:3-9. *See also* Kroeker, 3:9-16.

1.4.2 wherein the decompressed portion of boot data comprises a portion of the operating system.	KROCKER, as evidenced by the example citations below, discloses "wherein the decompressed portion of boot data comprises a portion of the operating system."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the decompressed portion of boot data comprises a portion of the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

2. The method of claim 1, wherein the updating comprises: associating additional boot data with the boot data list.	KROCKER, as evidenced by the example citations below, discloses “wherein the updating comprises: associating additional boot data with the boot data list.”

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein associating additional boot data with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.

KROCKER discloses this limitation:

See Claim 1.4.1 above.

See also

> “The prefetch table is updated to reflect disk location changes for the various records, or to reflect new records that were requested by the host computer but not found in cache during the previous power-on/reset.”

Kroeker, Abstract.

> “Additionally, the steps further include updating the data prefetch table, communicating the records from the disk to the host computer when the read counter exceeds a predetermined threshold, and setting a prefetch flag to inactive when the read counter exceeds the predetermined threshold.”

Kroeker, 2:59-64.

3. The method of claim 1, wherein the updating comprises: removing an association of additional boot data that is associated with the boot data list from the boot data list.	KROCKER, as evidenced by the example citations below, discloses “wherein the updating comprises: removing an association of additional boot data that is associated with the boot data list from the boot data list.”

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein removing an association of additional boot data that is associated with the boot data list from the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.

KROCKER discloses this limitation:

See Claim 1.4.1 above.

See also

> “The prefetch table is updated to reflect disk location changes for the various records, or to reflect new records that were requested by the host computer but not found in cache during the previous power-on/reset.”

Kroeker, Abstract.

> “Additionally, the steps further include updating the data prefetch table, communicating the records from the disk to the host computer when the read counter exceeds a predetermined threshold, and setting a prefetch flag to inactive when the read counter exceeds the predetermined threshold.”

Kroeker, 2:59-64.

4. The method of claim 1, wherein the updating comprises: associating additional boot data with the boot data list; and compressing a portion of the additional boot data.	KROCKER, as evidenced by the example citations below, discloses "wherein the updating comprises: associating additional boot data with the boot data list; and compressing a portion of the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein associating additional boot data with the boot data list; and compressing a portion of the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 1.4.1, and 2 above.	

5 (Preamble) A method for booting a computer system, the method comprising:	KROCKER, as evidenced by the example citations below, discloses "a method for booting a computer system, the method comprising:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

KROCKER discloses this limitation:

See Claims 1-1.4.2 above.

See also

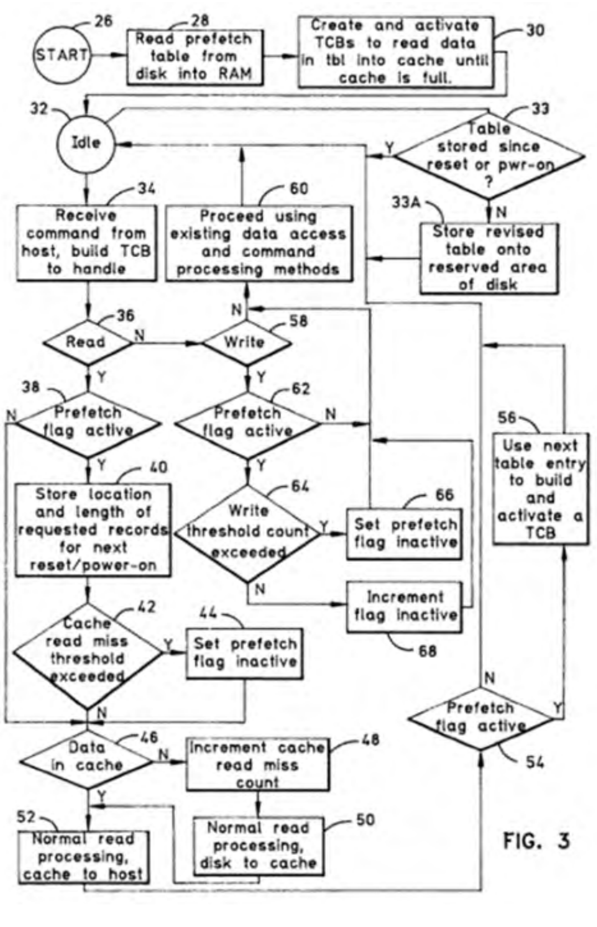

26
START
28
Read prefetch table from disk into RAM
Create and activate TCBs to read data in tbl into cache until cache is full.
30
32
Idle
33
Table stored since reset or pwr-on ?
Y
N
33A
Store revised table onto reserved area of disk
34
Receive command from host, build TCB to handle
60
Proceed using existing data access and command processing methods
36
Read
N
58
Write
Y
38
Prefetch flag active
62
Prefetch flag active
40
Store location and length of requested records for next reset/power-on
64
Write threshold count exceeded
66
Set prefetch flag inactive
Increment flag inactive
68
42
Cache read miss threshold exceeded
44
Set prefetch flag inactive
56
Use next table entry to build and activate a TCB
54
Prefetch flag active
46
Data in cache
48
Increment cache read miss count
52
Normal read processing, cache to host
50
Normal read processing, disk to cache

FIG. 3

Kroeker, Fig. 3.

> "A method for increasing boot speed of a host computer with associated hard disk drive generates a prefetch table that contains pointers to disk locations and lengths of the records of an application program requested by the host computer during an initial power-on/reset."

Kroeker, Abstract.

> "During the next power-on/reset, before the host computer is ready for data but after the disk drive has completed its reset routine, using the prefetch table the disk drive accesses the previously requested data and copies it onto the cache of the disk drive, from where it is transferred to the host computer when the host computer requests it."

Kroeker, Abstract.

> "Then, after a subsequent power-on or reset of the hard disk drive, and during a second power-on or reset of the host computer, the prefetch table is accessed to read into the data cache the data records."

Kroeker, 2:37-40.

> "As disclosed in detail below, these code means are for enhanced loading of the program from the hard disk drive to the host computer during power-on or reset of the host computer. In accordance with the present invention, code means receive a command from the host computer during a power-up or reset of the host computer."

Kroeker, 3:15-20.

> "Commencing at start state 26, the process moves to block 28, wherein a prefetch table is read from a reserved area of the disks 16 into the RAM cache 18."

Kroeker, 5:1-3.

> "In accordance with the present invention, steps executed by the digital processing apparatus include, after an initial power-up or reset of the hard disk drive and a host computer associated with the drive, receiving an initial read command from the host computer for transferring to the host computer a plurality of data records of a program stored on the disk. A prefetch table is then generated, with the table representing a disk location and length of each data record requested by the initial read command."

Kroeker, 2:20-37.

> "When the command is a read command, code means generate a prefetch table representative of at least the disk location of the records requested by the read command for transfer of the records from the disk to the cache for a subsequent power-on or reset of the host computer. Moreover, code means are provided for determining whether the records have been stored in the cache in response to a previous power-on or reset of the host computer, and the records are communicated to the host computer in response."

Kroeker, 3:21-29.

> "As discussed further below, the prefetch table contains a listing of the disk locations and lengths of data records that were requested by the host computer 14 in the immediately previous power-on/reset. Additionally, a copy of a prefetch flag, if enabled by the user, is created and set active at block 28."

Kroeker, 3:3-9. *See also* Kroeker, 3:9-16.

5.1 storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory;	KROCKER, as evidenced by the example citations below, discloses "storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 1.1 above. *See also* "During the idle state 32, the process can move to decision diamond 33 to determine whether the prefetch table has been stored since the latest power on or reset" Kroeker, 5:22-24.	

5.2 loading the stored compressed boot data from the first memory;	KROCKER, as evidenced by the example citations below, discloses "loading the stored compressed boot data from the first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the stored compressed boot data from the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 1.1 above.	

5.3 accessing the loaded compressed boot data;	KROCKER, as evidenced by the example citations below, discloses "accessing the loaded compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 1.2 above.	

5.4 decompressing the accessed compressed boot data;	KROCKER, as evidenced by the example citations below, discloses "decompressing the accessed compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 1.3 above.	

5.5 utilizing the decompressed boot data to at least partially boot the computer system;	KROCKER, as evidenced by the example citations below, discloses "utilizing the decompressed boot data to at least partially boot the computer system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed boot data to at least partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 1.3 and 1.4.2 above.	

5.6 updating the boot data list;	KROCKER, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 1.4.1 above.	

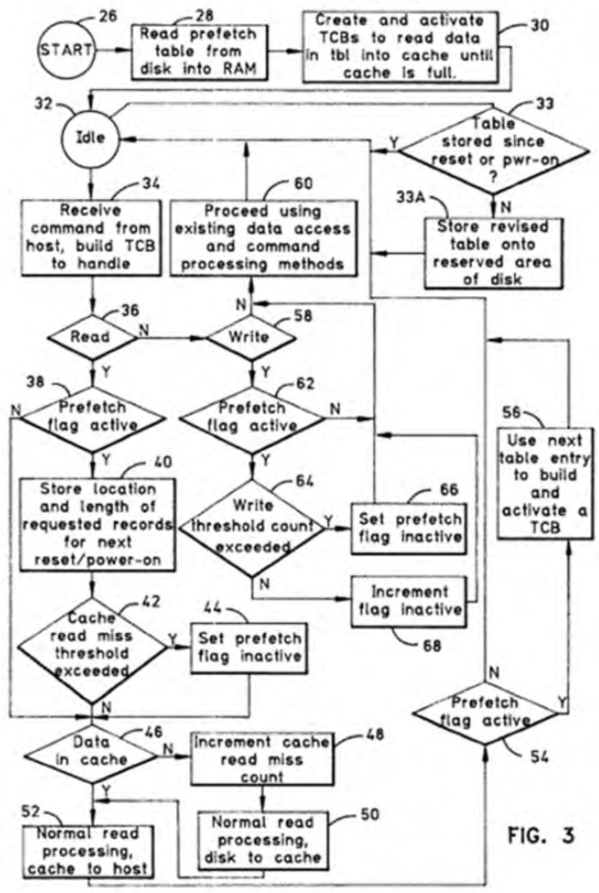
5.7 wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form.	KROCKER, as evidenced by the example citations below, discloses "wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 1-1.3 above.	

6 (Preamble) A system comprising: a processor;	KROCKER, as evidenced by the example citations below, discloses "a system comprising: a processor:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system comprising: a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

KROCKER discloses this limitation:

See ***<u>Add disclosure from '608 Patent Claim 1.2</u>***


START
26
Read prefetch table from disk into RAM
28
Create and activate TCBs to read data in tbl into cache until cache is full.
30
Idle
32
Table stored since reset or pwr-on ?
33
Y
N
Store revised table onto reserved area of disk
33A
Receive command from host, build TCB to handle
34
Proceed using existing data access and command processing methods
60
Read
36
Write
58
Prefetch flag active
38
Prefetch flag active
62
Store location and length of requested records for next reset/power-on
40
Write threshold count exceeded
64
Set prefetch flag inactive
66
Increment flag inactive
68
Use next table entry to build and activate a TCB
56
Cache read miss threshold exceeded
42
Set prefetch flag inactive
44
Prefetch flag active
54
Data in cache
46
Increment cache read miss count
48
Normal read processing, disk to cache
50
Normal read processing, cache to host
52
FIG. 3

Kroeker, Fig. 3.

> "During the next power-on/reset, before the host computer is ready for data but after the disk drive has completed its reset routine, using the

> prefetch table the disk drive accesses the previously requested data and copies it onto the cache of the disk drive, from where it is transferred to the host computer when the host computer requests it."

Kroeker, Abstract.

> "Then, after a subsequent power-on or reset of the hard disk drive, and during a second power-on or reset of the host computer, the prefetch table is accessed to read into the data cache the data records."

Kroeker, 2:37-40.

> "As disclosed in detail below, these code means are for enhanced loading of the program from the hard disk drive to the host computer during power-on or reset of the host computer. In accordance with the present invention, code means receive a command from the host computer during a power-up or reset of the host computer."

Kroeker, 3:15-20.

> "Commencing at start state 26, the process moves to block 28, wherein a prefetch table is read from a reserved area of the disks 16 into the RAM cache 18."

Kroeker, 5:1-3.

6.1 a memory; and	KROCKER, as evidenced by the example citations below, discloses “a memory”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a memory.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 1.1 and 1.2 above.	

6.2 a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor	KROCKER, as evidenced by the example citations below, discloses "a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

KROCKER discloses this limitation:

See Claims 1.1 and 1.2 above.

See ***Add disclosure from '608 Patent Claim 1.2***

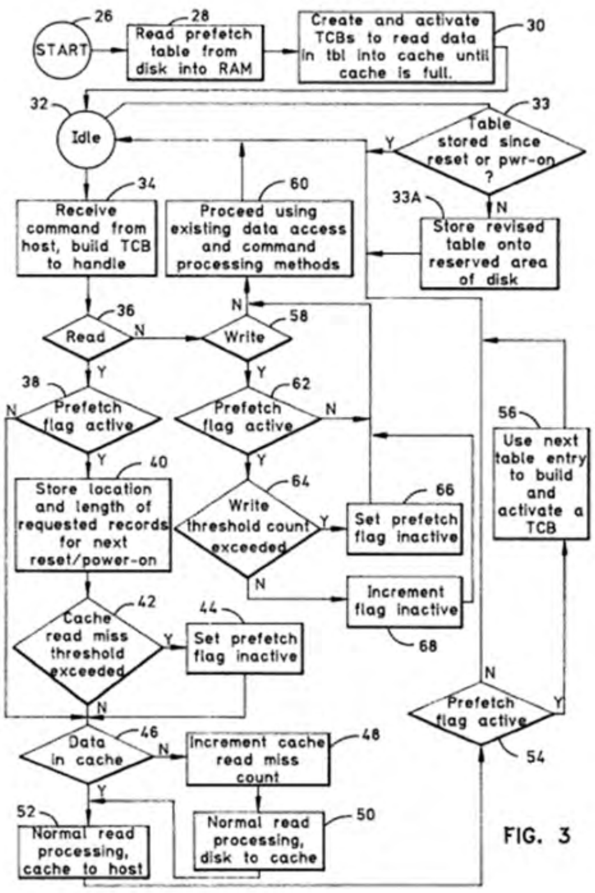

START
26
28
Read prefetch table from disk into RAM
Create and activate TCBs to read data in tbl into cache until cache is full.
30
32
Idle
33
Table stored since reset or pwr-on ?
Y
N
33A
Store revised table onto reserved area of disk
34
Receive command from host, build TCB to handle
60
Proceed using existing data access and command processing methods
36
Read
N
Write
58
Y
38
Prefetch flag active
62
56
Use next table entry to build and activate a TCB
40
Store location and length of requested records for next reset/power-on
64
Write threshold count exceeded
66
Set prefetch flag inactive
Increment flag inactive
68
42
Cache read miss threshold exceeded
44
Set prefetch flag inactive
54
46
Data in cache
48
Increment cache read miss count
52
Normal read processing, cache to host
50
Normal read processing, disk to cache

FIG. 3

Kroeker, Fig. 3.

> "During the next power-on/reset, before the host computer is ready for data but after the disk drive has completed its reset routine, using the prefetch table the disk drive accesses the previously requested data and copies it onto the cache of the disk drive, from where it is transferred to the host computer when the host computer requests it."

Kroeker, Abstract.

> "Then, after a subsequent power-on or reset of the hard disk drive, and during a second power-on or reset of the host computer, the prefetch table is accessed to read into the data cache the data records."

Kroeker, 2:37-40.

> "As disclosed in detail below, these code means are for enhanced loading of the program from the hard disk drive to the host computer during power-on or reset of the host computer. In accordance with the present invention, code means receive a command from the host computer during a power-up or reset of the host computer."

Kroeker, 3:15-20.

> "Commencing at start state 26, the process moves to block 28, wherein a prefetch table is read from a reserved area of the disks 16 into the RAM cache 18."

Kroeker, 5:1-3.

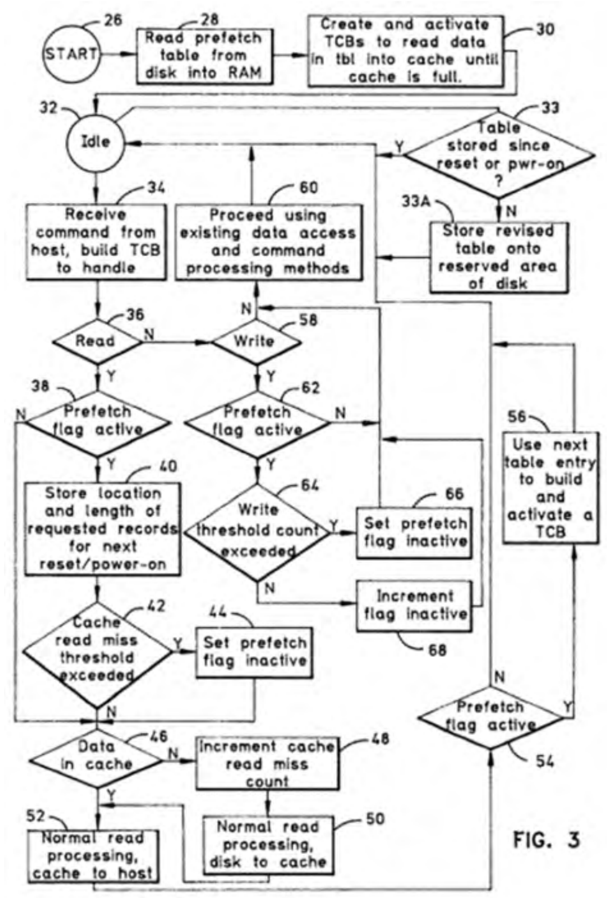
Appendix C11
Invalidity of U.S. Patent 8,880,862 based on Krocker

6.3 wherein the processor is configured:	KROCKER, as evidenced by the example citations below, discloses “wherein the processor is configured:”

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the processor is configured), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.

KROCKER discloses this limitation:

See



START
26
Read prefetch table from disk into RAM
28
Create and activate TCBs to read data in tbl into cache until cache is full.
30
Idle
32
Table stored since reset or pwr-on ?
33
Y
N
33A
Store revised table onto reserved area of disk
Receive command from host, build TCB to handle
34
Proceed using existing data access and command processing methods
60
Read
36
N
Write
58
Y
Prefetch flag active
38
Prefetch flag active
62
Store location and length of requested records for next reset/power-on
40
Write threshold count exceeded
64
Set prefetch flag inactive
66
Increment flag inactive
68
Cache read miss threshold exceeded
42
Set prefetch flag inactive
44
Use next table entry to build and activate a TCB
56
Prefetch flag active
54
Data in cache
46
Increment cache read miss count
48
Normal read processing, cache to host
52
Normal read processing, disk to cache
50

FIG. 3

Kroeker, Fig. 3.

> “During the next power-on/reset, before the host computer is ready for data but after the disk drive has completed its reset routine, using the

> prefetch table the disk drive accesses the previously requested data and copies it onto the cache of the disk drive, from where it is transferred to the host computer when the host computer requests it."

Kroeker, Abstract.

> "Then, after a subsequent power-on or reset of the hard disk drive, and during a second power-on or reset of the host computer, the prefetch table is accessed to read into the data cache the data records."

Kroeker, 2:37-40.

> "As disclosed in detail below, these code means are for enhanced loading of the program from the hard disk drive to the host computer during power-on or reset of the host computer. In accordance with the present invention, code means receive a command from the host computer during a power-up or reset of the host computer."

Kroeker, 3:15-20.

> "Commencing at start state 26, the process moves to block 28, wherein a prefetch table is read from a reserved area of the disks 16 into the RAM cache 18."

Kroeker, 5:1-3.

6.4 to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory,	KROCKER, as evidenced by the example citations below, discloses "to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 1.1 above.	

6.5 to access the loaded portion of the boot data in the compressed form,	KROCKER, as evidenced by the example citations below, discloses "to access the loaded portion of the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to access the loaded portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 1.2 above.	

6.6 to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data, and	KROCKER, as evidenced by the example citations below, discloses "to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 1.3 above.	

6.7 to update the boot data list.	KROCKER, as evidenced by the example citations below, discloses "to update the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to update the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 1.4.1 above.	

Appendix C11
Invalidity of U.S. Patent 8,880,862 based on Krocker

8 (Preamble) A method of loading an operating system for booting a computer system, comprising:	KROCKER, as evidenced by the example citations below, discloses "A method of loading an operating system for booting a computer system, comprising:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method of loading an operating system for booting a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 1 (Preamble) above.	

Appendix C11
Invalidity of U.S. Patent 8,880,862 based on Krocker

8.1 storing a portion of the operating system in a compressed form in a first memory;	KROCKER, as evidenced by the example citations below, discloses "storing a portion of the operating system in a compressed form in a first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing a portion of the operating system in a compressed form in a first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 6.2 above.	

8.2 loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list;	KROCKER, as evidenced by the example citations below, discloses "loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 1.1 above.	

8.3 accessing the loaded portion of the operating system from the second memory in the compressed form;	KROCKER, as evidenced by the example citations below, discloses "accessing the loaded portion of the operating system from the second memory in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded portion of the operating system from the second memory in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 1.2 above.	

8.4 decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system;	KROCKER, as evidenced by the example citations below, discloses "decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 1.3 and 1.4.2 above.	

8.5 utilizing the decompressed portion of the operating system to at least partially boot the computer system; and	KROCKER, as evidenced by the example citations below, discloses “utilizing the decompressed portion of the operating system to at least partially boot the computer system”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed portion of the operating system to at least partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 1.3 and 1.4.2 above.	

8.6 updating the boot data list,	KROCKER, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 1.4.1 above.	

8.7 wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form.	KROCKER, as evidenced by the example citations below, discloses "wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 1-1.3 and 5.7 above.	

9.1 The method of claim 8, further comprising: compressing an additional portion of the operating system that is not associated with the boot data list; and	KROCKER, as evidenced by the example citations below, discloses “the method of claim 8, further comprising: compressing an additional portion of the operating system that is not associated with the boot data list”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, compressing an additional portion of the operating system that is not associated with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 1.1 above.	

9.2 storing the additional portion of the operating system in the first memory, and	KROCKER, as evidenced by the example citations below, discloses "storing the additional portion of the operating system in the first memory"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing the additional portion of the operating system in the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

KROCKER discloses this limitation:

See Claim 8.1 above.

> "The prefetch table is updated to reflect disk location changes for the various records, or to reflect new records that were requested by the host computer but not found in cache during the previous power-on/reset."

Kroeker, Abstract.

> "Additionally, the steps further include updating the data prefetch table, communicating the records from the disk to the host computer when the read counter exceeds a predetermined threshold, and setting a prefetch flag to inactive when the read counter exceeds the predetermined threshold."

Kroeker, 2:59-64.

9.3 wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system.	KROCKER, as evidenced by the example citations below, discloses "wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

KROCKER discloses this limitation:

See Claim 8.5 above.

> "The prefetch table is updated to reflect disk location changes for the various records, or to reflect new records that were requested by the host computer but not found in cache during the previous power-on/reset."

Kroeker, Abstract.

> "Additionally, the steps further include updating the data prefetch table, communicating the records from the disk to the host computer when the read counter exceeds a predetermined threshold, and setting a prefetch flag to inactive when the read counter exceeds the predetermined threshold."

Kroeker, 2:59-64.

10. The method of claim 9, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 9, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 9.1 above.	

Appendix C11
Invalidity of U.S. Patent 8,880,862 based on Krocker

11 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	KROCKER, as evidenced by the example citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, comprising:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 1 (Preamble) above.	

Appendix C11
Invalidity of U.S. Patent 8,880,862 based on Krocker

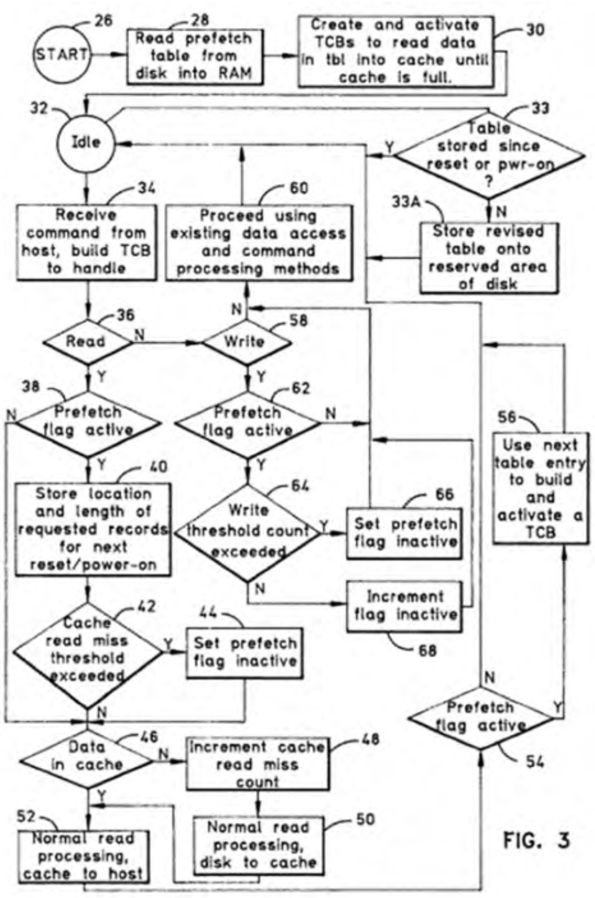
11.1 loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system;	KROCKER, as evidenced by the example citations below, discloses “loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system”

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.

KROCKER discloses this limitation:

See Claim 1.1 above.

See also



START
26
Read prefetch table from disk into RAM
28
Create and activate TCBs to read data in tbl into cache until cache is full.
30
32
Idle
Table stored since reset or pwr-on ?
33
Y
N
33A
Store revised table onto reserved area of disk
34
Receive command from host, build TCB to handle
60
Proceed using existing data access and command processing methods
36
Read
N
Write
58
Y
38
Prefetch flag active
N
62
Prefetch flag active
Y
40
Store location and length of requested records for next reset/power-on
64
Write threshold count exceeded
66
Set prefetch flag inactive
Increment flag inactive
68
56
Use next table entry to build and activate a TCB
42
Cache read miss threshold exceeded
44
Set prefetch flag inactive
Prefetch flag active
54
46
Data in cache
48
Increment cache read miss count
52
Normal read processing, cache to host
50
Normal read processing, disk to cache

FIG. 3

Kroeker, Fig. 3.

> "During the next power-on/reset, before the host computer is ready for data but after the disk drive has completed its reset routine, using the prefetch table the disk drive accesses the previously requested data and copies it onto the cache of the disk drive, from where it is transferred to the host computer when the host computer requests it."

Kroeker, Abstract.

> "Then, after a subsequent power-on or reset of the hard disk drive, and during a second power-on or reset of the host computer, the prefetch table is accessed to read into the data cache the data records."

Kroeker, 2:37-40.

> "As disclosed in detail below, these code means are for enhanced loading of the program from the hard disk drive to the host computer during power-on or reset of the host computer. In accordance with the present invention, code means receive a command from the host computer during a power-up or reset of the host computer."

Kroeker, 3:15-20.

> "Commencing at start state 26, the process moves to block 28, wherein a prefetch table is read from a reserved area of the disks 16 into the RAM cache 18."

Kroeker, 5:1-3.

> "If the command is a read command, indicating that the host computer 14, pursuant to its initialization, is requesting data records that are part of a computer program such as DOS or Windows, the process moves to decision diamond 38 wherein it is determined whether the prefetch flag is active."

Kroeker, 5:41-46. *See also* Kroeker, 5:47-6:12.

11.2 accessing the loaded boot data in compressed form from the memory;	KROCKER, as evidenced by the example citations below, discloses "accessing the loaded boot data in compressed form from the memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded boot data in compressed form from the memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 1.2 above.	

Appendix C11
Invalidity of U.S. Patent 8,880,862 based on Krocker

11.3 decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form;	KROCKER, as evidenced by the example citations below, discloses "decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 1.3 above.	

11.4 utilizing the decompressed boot data to load at least a portion of the operating system for the computer system; and	KROCKER, as evidenced by the example citations below, discloses "utilizing the decompressed boot data to load at least a portion of the operating system for the computer system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed boot data to load at least a portion of the operating system for the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

11.5 updating the boot data list.	KROCKER, as evidenced by the example citations below, discloses “updating the boot data list”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 1.4.1 above.	

13 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	KROCKER, as evidenced by the example citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, comprising"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 1 (Preamble) above.	

13.1 loading boot data in a compressed form that is associated with a boot data list from a boot device;	KROCKER, as evidenced by the example citations below, discloses "loading boot data in a compressed form that is associated with a boot data list from a boot device"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading boot data in a compressed form that is associated with a boot data list from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 1.1 above.	

13.2 accessing the loaded boot data in the compressed form;	KROCKER, as evidenced by the example citations below, discloses "accessing the loaded boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 1.2 above.	

13.3 decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form;	KROCKER, as evidenced by the example citations below, discloses "decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 1.3 above.	

13.4 updating the boot data list.	KROCKER, as evidenced by the example citations below, discloses “updating the boot data list”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 1.4.1 above.	

Appendix C11
Invalidity of U.S. Patent 8,880,862 based on Krocker

14 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	KROCKER, as evidenced by the example citations below, discloses “a method for providing accelerated loading of an operating system in a computer system, comprising”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 1 (Preamble) above.	

Appendix C11
Invalidity of U.S. Patent 8,880,862 based on Krocker

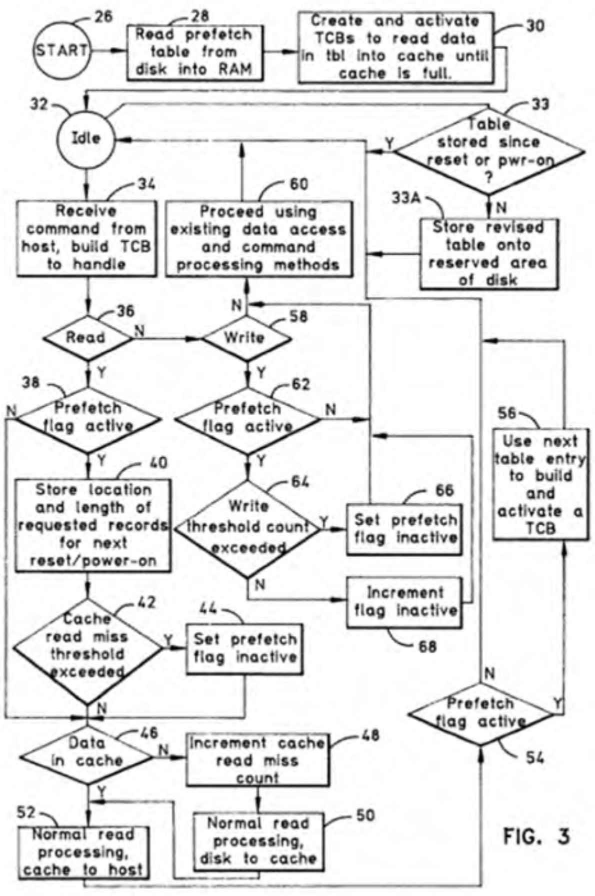
14.1 accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list;	KROCKER, as evidenced by the example citations below, discloses "accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

KROCKER discloses this limitation:

See Claims 1.1 and 1.2 above.

See



START
26
Read prefetch table from disk into RAM
28
Create and activate TCBs to read data in tbl into cache until cache is full.
30
32
Idle
Table stored since reset or pwr-on ?
33
Y
N
33A
Store revised table onto reserved area of disk
34
Receive command from host, build TCB to handle
60
Proceed using existing data access and command processing methods
36
Read
N
Write
58
N
Y
38
Prefetch flag active
N
Y
62
Prefetch flag active
N
Y
40
Store location and length of requested records for next reset/power-on
64
Write threshold count exceeded
Y
N
66
Set prefetch flag inactive
Increment flag inactive
68
56
Use next table entry to build and activate a TCB
42
Cache read miss threshold exceeded
Y
N
44
Set prefetch flag inactive
Prefetch flag active
N
Y
54
46
Data in cache
N
Y
48
Increment cache read miss count
50
Normal read processing, disk to cache
52
Normal read processing, cache to host

FIG. 3

Kroeker, Fig. 3.

> "A method for increasing boot speed of a host computer with associated hard disk drive generates a prefetch table that contains pointers to disk locations and lengths of the records of an application program requested by the host computer during an initial power-on/reset."

Kroeker, Abstract.

> "In accordance with the present invention, steps executed by the digital processing apparatus include, after an initial power-up or reset of the hard disk drive and a host computer associated with the drive, receiving an initial read command from the host computer for transferring to the host computer a plurality of data records of a program stored on the disk. A prefetch table is then generated, with the table representing a disk location and length of each data record requested by the initial read command."

Kroeker, 2:20-37.

> "When the command is a read command, code means generate a prefetch table representative of at least the disk location of the records requested by the read command for transfer of the records from the disk to the cache for a subsequent power-on or reset of the host computer. Moreover, code means are provided for determining whether the records have been stored in the cache in response to a previous power-on or reset of the host computer, and the records are communicated to the host computer in response."

Kroeker, 3:21-29.

> "As discussed further below, the prefetch table contains a listing of the disk locations and lengths of data records that were requested by the host computer 14 in the immediately previous power-on/reset. Additionally, a copy of a prefetch flag, if enabled by the user, is created and set active at block 28."

Kroeker, 3:3-9. *See also* Kroeker, 3:9-16.

14.2 loading the boot data into a memory; and	KROCKER, as evidenced by the example citations below, discloses "loading the boot data into a memory"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the boot data into a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

KROCKER discloses this limitation:

See Claim 1.1 above.

14.3 servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form; and	KROCKER, as evidenced by the example citations below, discloses "servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

KROCKER discloses this limitation:

See Claims 1.2 and 1.3 above.

See also

> "Then, after a subsequent power-on or reset of the hard disk drive, and during a second power-on or reset of the host computer, the prefetch table is accessed to read into the data cache the data records. In response to a subsequent read command from the host computer, it is determined whether records requested by the subsequent read command are stored in the data cache. If they are, the records are communicated from the cache to the host computer; otherwise, the records are communicated from the disk to the host computer."

Kroeker, 2:37-46.

> "Preferably, the accessing and determining steps are repeated for each power-on or reset of the host computer."

Kroeker, 2:47-48.

> "Additionally, the disk drive includes means for communicating the data from the cache to the host computer during a second hardware reset of

> the host computer."

Kroeker, 3:36-38.

> "Additionally, as intended by the present invention the onboard controller 20 includes an adaptive cache module 24. Per the present invention, the adaptive cache module 24 is executed by the onboard controller 20 as a series of computer-executable instructions."

Kroeker, 4:16-20.

> "If it is active, the logic, at block 56, uses the next entry in the prefetch table to build a task control block (TCB) to fetch data into the same segment of the cache 18 that the just-transferred record had occupied prior to being communicated to the host computer 14. In accordance with the present invention, the TCB in block 50 is activated as though a command otherwise was received across the device/file interface. In other words, when the host computer 14 is a PC, the TCB in block 50 is activated as though a command otherwise was received across the SCSI (or IDE)--disk drive interface. In this way, the relatively small amount of cache storage space can be optimally used during the adaptive caching process until all records designated in the prefetch table have been loaded into cache and then transferred to the host computer 14."

Kroeker, 6:16-31.

> "If the command is not a write command, the process moves to block 60 to proceed using existing data access and command processing methods, and then the process continues back to the idle state 32."

Kroeker, 6:44-48.

14.4 updating the boot data list.	KROCKER, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 1.4.1 above.	

Appendix C11
Invalidity of U.S. Patent 8,880,862 based on Krocker

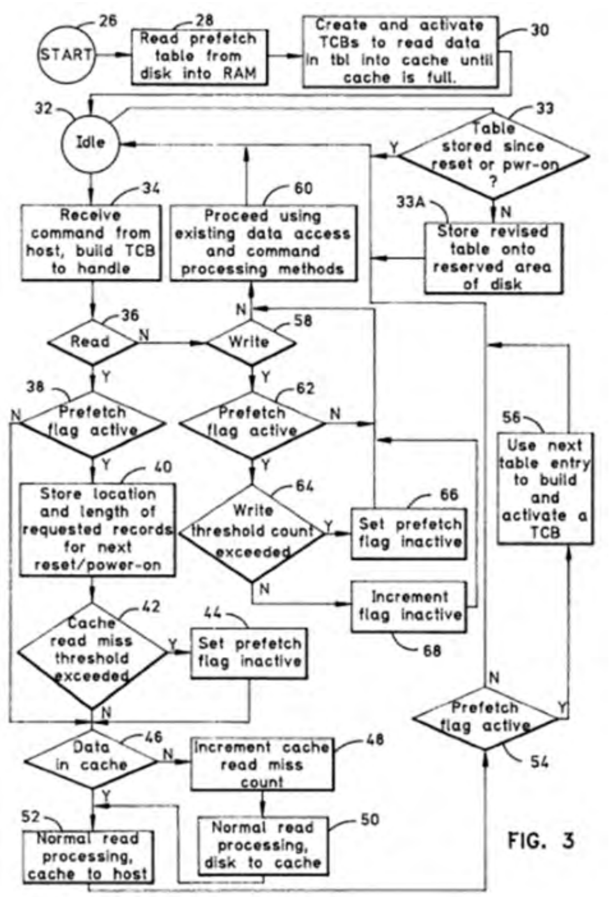
15. The method of claim 14, wherein the boot data comprises: a program code associated with the operating system.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 14, wherein the boot data comprises: a program code associated with the operating system."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the boot data comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

KROCKER discloses this limitation:

See Claims 1 (Preamble) and 1.1 above.

See



START
26
Read prefetch table from disk into RAM
28
Create and activate TCBs to read data in tbl into cache until cache is full.
30
Idle
32
Table stored since reset or pwr-on ?
33
Y
N
Store revised table onto reserved area of disk
33A
Receive command from host, build TCB to handle
34
Proceed using existing data access and command processing methods
60
Read
36
Write
58
Prefetch flag active
38
Prefetch flag active
62
Store location and length of requested records for next reset/power-on
40
Write threshold count exceeded
64
Set prefetch flag inactive
66
Increment flag inactive
68
Use next table entry to build and activate a TCB
56
Cache read miss threshold exceeded
42
Set prefetch flag inactive
44
Prefetch flag active
54
Data in cache
46
Increment cache read miss count
48
Normal read processing, cache to host
52
Normal read processing, disk to cache
50

FIG. 3

Kroeker, Fig. 3.

> "A method for increasing boot speed of a host computer with associated hard disk drive generates a prefetch table that contains pointers to disk locations and lengths of the records of an application program requested by the host computer during an initial power-on/reset."

Kroeker, Abstract.

> "And, the hard disk drive 12 can be any hard disk drive suitable for computer applications, provided that the hard disk drive 12 includes at least one, and typically a plurality of, data storage disks 16 and an on-board, solid state, random access memory (RAM) data cache 18."

Kroeker, 4:5:10.

16. The method of claim 14, wherein the operating system comprises: a plurality of files.	KROCKER, as evidenced by the example citations below, discloses “the method of claim 14, wherein the operating system comprises: a plurality of files.”

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.

KROCKER discloses this limitation:

See

> “It happens that the transfer of the selected program to the computer is relatively slow, particularly when the program is a modern large operating system.”

Kroeker, 1:29-31.

17. The method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 15 above.	

19.1 The method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:	KROCKER, as evidenced by the example citations below, discloses "the method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

19.2 associating the accessed boot data that is not associated with the boot data list to the boot data list.	KROCKER, as evidenced by the example citations below, discloses "associating the accessed boot data that is not associated with the boot data list to the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, associating the accessed boot data that is not associated with the boot data list to the boot data list.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 2 above.	

23. The method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 16 above.	

24. The method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system.
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 15 above.	

27. The method of claim 1, wherein the memory comprises: a physical memory.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 1, wherein the memory comprises: a physical memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 1.1 and 1.2 above.	

28. The method of claim 1, wherein the operating system comprises: a plurality of files.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 1, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 16 above.	

29. The method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 15 and 17 above.	

31. The method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 1.1 and 1.2 above.	

Appendix C11
Invalidity of U.S. Patent 8,880,862 based on Krocker

32. The method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

33. The method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

35. The method of claim 5, wherein the compressed boot data represents a plurality of files.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 5, wherein the compressed boot data represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the compressed boot data represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 16 and 23 above.	

36. The method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 15 and 17 above.	

39. The method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 1.1 and 1.2 above.	

40. The method of claim 36, wherein the operating system comprises: a plurality of files.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 36, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 36, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 16 and 23 above.	

43. The method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 31 above.	

44. The method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 32 above.	

45. The method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 32 and 33 above.	

47. The system of claim 6, wherein the boot data in the compressed form represents a plurality of files.	KROCKER, as evidenced by the example citations below, discloses “the system of claim 6, wherein the boot data in the compressed form represents a plurality of files”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 16 and 23 above.	

48. The system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system.	KROCKER, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 15, 17 and 24 above.	

49. The system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form.	KROCKER, as evidenced by the example citations below, discloses "the system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 10 above.	

Appendix C11
Invalidity of U.S. Patent 8,880,862 based on Krocker

50. The system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form.	KROCKER, as evidenced by the example citations below, discloses "the system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

51. The system of claim 6, wherein the first memory comprises: a physical memory.	KROCKER, as evidenced by the example citations below, discloses "the system of claim 6, wherein the first memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the first memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 27 and 39 above.	

52. The system of claim 6, wherein the boot data the compressed form comprises: a plurality of files.	KROCKER, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data the compressed form comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data the compressed form comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 16 and 23 above.	

53. The system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program.	KROCKER, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 15, 17 and 24 above.	

59. The method of claim 8, wherein the operating system in the compressed form represents a plurality of files.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 16 and 23 above.	

60. The method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system.	KROCKER, as evidenced by the example citations below, discloses “the method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 15, 17 and 24 above.	

63. The method of claim 8, wherein the second memory comprises: a physical memory.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 8, wherein the second memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the second memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 27 and 38 above.	

64. The method of claim 8, wherein the operating system comprises: a plurality of files.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 16 and 23 above.	

65. The method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 15, 17 and 24 above.	

67. The method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 31 above.	

71. The method of claim 11, wherein the boot data in the compressed form represents a plurality of files.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 11, wherein the boot data in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 16 and 23 above.	

Appendix C11
Invalidity of U.S. Patent 8,880,862 based on Krocker

72. The method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system.	KROCKER, as evidenced by the example citations below, discloses “the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 15, 17 and 24 above.	

75. The method of claim 11, wherein the memory comprises: a physical memory.	KROCKER, as evidenced by the example citations below, discloses “the method of claim 11, wherein the memory comprises: a physical memory”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 27 above.	

76. The method of claim 11, wherein the operating system comprises: a plurality of files.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 11, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 16 and 23 above.	

77. The method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 15, 17 and 24 above.	

79. The method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 31 above.	

80. The method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 32, 33 and 49 above.	

81. The method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 32, 33 and 49 above.	

83. The method of claim 13, wherein the boot data in the compressed form represents a plurality of files.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 13, wherein the boot data in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 16 and 23 above.	

84. The method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 15, 17 and 24 above.	

87. The method of claim 13, wherein the memory comprises: a physical memory.	KROCKER, as evidenced by the example citations below, discloses “the method of claim 13, wherein the memory comprises: a physical memory”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 27 above.	

88. The method of claim 13, wherein the operating system comprises: a plurality of files.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 13, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 16 and 23 above.	

89. The method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 15, 17 and 24 above.	

91. The method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 31 above.	

92. The method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 32, 33 and 49 above.	

93. The method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claim 32, 33, and 49 above.	

97.1 The method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list,	KROCKER, as evidenced by the example citations below, discloses "the method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 9.1-9.3 and 19 above.	

97.2 and wherein the updating comprises: associating the additional compressed boot data with the boot data list.	KROCKER, as evidenced by the example citations below, discloses "and wherein the updating comprises: associating the additional compressed boot data with the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, and wherein the updating comprises: associating the additional compressed boot data with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 1.4.1, 2, 4, 5.6, and 8.6 above.	

98. The method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 1.4.1, 5.6, and 8.6 above.	

107. The method of claim 2, further comprising: storing the updated boot list in a non-volatile memory.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 2, further comprising: storing the updated boot list in a non-volatile memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 2, further comprising: storing the updated boot list in a non-volatile memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 1.4.1, 5.6, and 8.6 above.	

108. The method of claim 2, further comprising: compressing at least a portion of the additional boot data.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 2, further comprising: compressing at least a portion of the additional boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 2, further comprising: compressing at least a portion of the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 1.1, 5, 9.1 and 8 above.	

109. The method of claim 108, further comprising: storing the compressed additional boot data.	KROCKER, as evidenced by the example citations below, discloses "the method of claim 108, further comprising: storing the compressed additional boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 108, further comprising: storing the compressed additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. KROCKER discloses this limitation: *See* Claims 5.1, 8.1 and 9.1 above.	

Appendix C12
Invalidity of U.S. Patent 8,880,862 based on Lee

U.S. Patent Application Publication No. 2001/0039612 Lee ("Lee") invalidates claims 1-6, 8-11, 13-17, 19, 23-24, 27-29, 31-33, 35-36, 39-40, 43-45, 47-53, 59-60, 63-65, 67, 71-72, 75-77, 79-81, 83-84, 87-89, 91-93, 97-98, and 107-109 of United States Patent No. 8,880,862 ("the '862 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

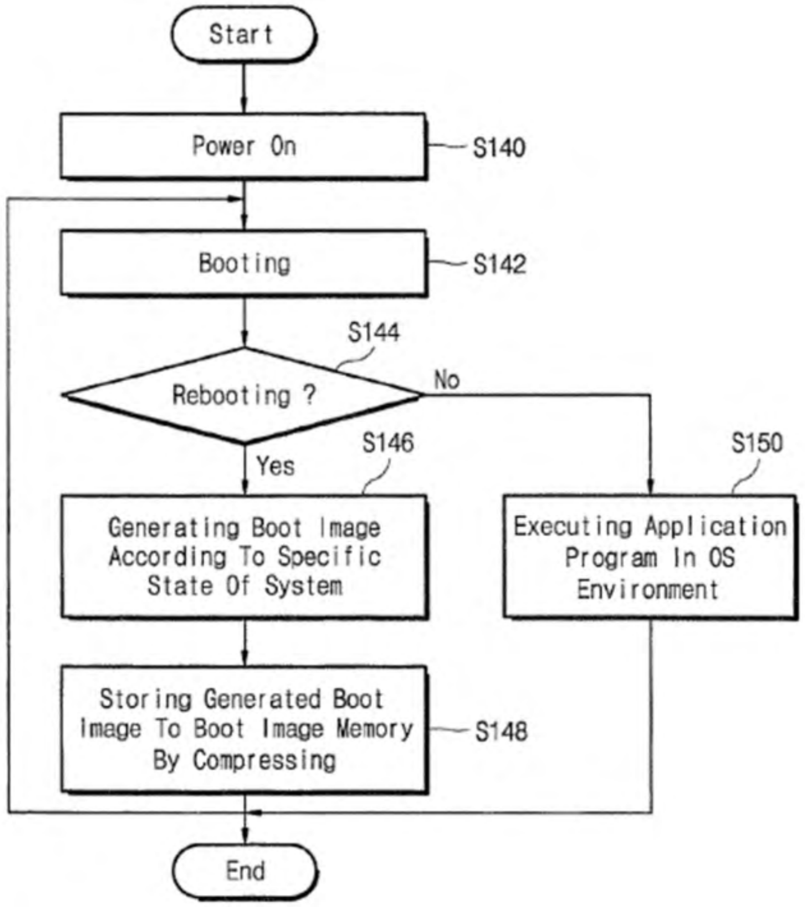
The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '862 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix C12
Invalidity of U.S. Patent 8,880,862 based on Lee

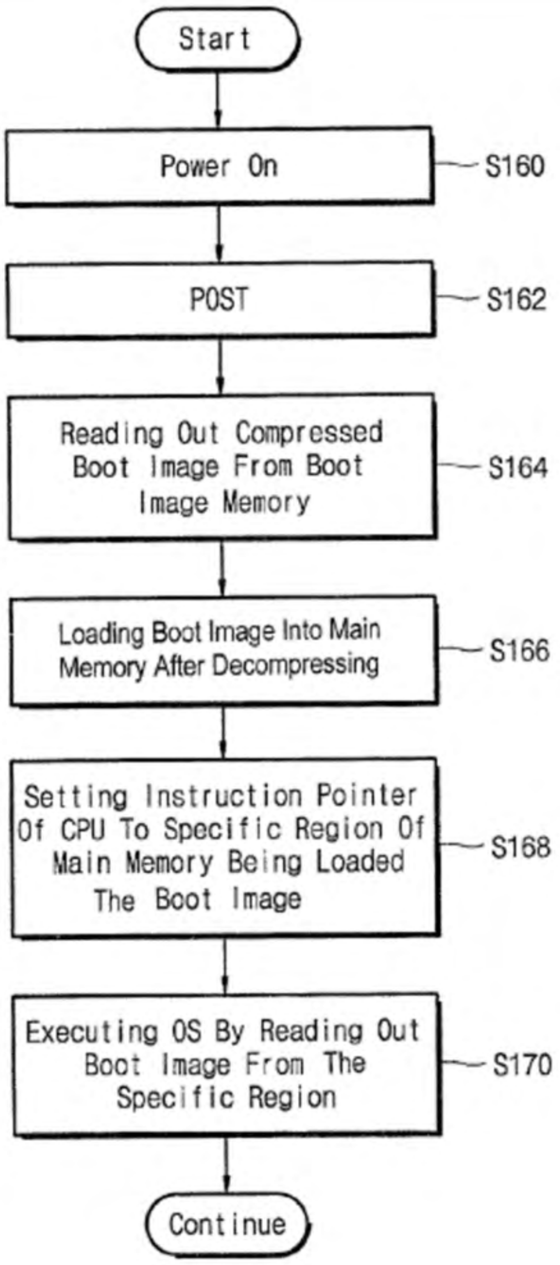
1 (Preamble) A method for providing accelerated loading of an operating system in a computer system, the method comprising:	Lee, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, the method comprising:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system in a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lee discloses this limitation:


Start
Power On
S140
Booting
S142
S144
Rebooting ?
No
Yes
S146
S150
Generating Boot Image According To Specific State Of System
Executing Application Program In OS Environment
Storing Generated Boot Image To Boot Image Memory By Compressing
S148
End

Lee, Fig. 2.


Start
Power On
S160
POST
S162
Reading Out Compressed Boot Image From Boot Image Memory
S164
Loading Boot Image Into Main Memory After Decompressing
S166
Setting Instruction Pointer Of CPU To Specific Region Of Main Memory Being Loaded The Boot Image
S168
Executing OS By Reading Out Boot Image From The Specific Region
S170
Continue

Lee, Fig. 3.

> "Assuming that the correct first diskette is inserted into the system, the ROM-based code retrieves the master boot record from sector 0 of the first diskette."

Lee, ¶ 0014.

> "The ROM-based code then copies the OS loader into RAM from the first

> diskette, starting at the address indicated in the master boot record and continuing for a length indicated by the offset provided by the master boot record. After copying the OS loader into RAM, the ROM-based code jumps to the OS loader."

Lee, ¶ 0014.

> "The OS loader is more sophisticated than the ROM-based code and performs certain preliminary functions, such as sizing memory. After performing preliminary functions, the OS loader copies into RAM a portion of the operating system known as the "kernel.""

Lee, ¶ 0015.

> "It is therefore an object of the present invention to provide a computer system to reduce booting time."

Lee, ¶ 0023.

> It is another object of the invention to provide a method for shutting off and booting a computer system to reduce booting time."

Lee, ¶ 0024.

> "The BIOS ROM 106 and the boot image memory 108 are capable of setting and storing an initial state of main memory by a manufacturer or a user. Thus, the CPU 102 can reduce a device drive loading time by reading out the compressed boot image from the boot image memory 108 and loading the boot image after decompressing, when boot image is loaded to the main memory 104."

Lee, ¶ 0040.

> "In other words, at the step S142, the CPU 102 executes the operating system if a successful boot is detected through a POST routine. Therefore, a certain application program can be executed in the operating system environment."

Lee, ¶ 0042.

1.1 loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory;	Lee, as evidenced by the example citations below, discloses “loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory;”

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.

Lee discloses this limitation:

> “The ROM-based code attempts to establish communication with a so-called “boot device”. A boot device holds information that is necessary to boot the system. In attempting to establish communication with a boot device, the ROM-based code operates according to a so-called “boot order.””

Lee, ¶ 0012.

> “More particularly, the ROM-based code attempts to retrieve a so-called “master boot record” from a particular sector of the diskette. If the communication attempt is successful, the ROM-based code uses that device as the boot device. If not, the ROM-based code proceeds to attempt communication With a device of the next boot order, e.g., local media.”

Lee, ¶ 0012.

> “Assuming that the correct first diskette is inserted into the system, the ROM-based code retrieves the master boot record from sector 0 of the first diskette.”

Lee, ¶ 0014.

> “The ROM-based code then copies the OS loader into RAM from the first diskette, starting at the address indicated in the master boot record and continuing for a length indicated by the offset provided by the master boot record. After copying the OS loader into RAM, the ROM-based code

jumps to the OS loader."

Lee, ¶ 0014.

> "The OS loader is more sophisticated than the ROM-based code and performs certain preliminary functions, such as sizing memory. After performing preliminary functions, the OS loader copies into RAM a portion of the operating system known as the "kernel.""

Lee, ¶ 0015.

> "In order to attain the above objects, according to an aspect of the present invention, there is provided a computer system having a central processing unit; a main and/or auxiliary power supply for supplying main and/or auxiliary power of the computer system; a boot image storing device for storing a boot image of the computer system; a main memory for storing the boot image from the boot image storing device by receiving the auxiliary power when the main power is shut off; and a composition memory for setting an instruction pointer of the central processing unit to a specific region of the main memory storing the boot image; wherein the central processing unit loads the boot image from the specific region of the main memory in response to the instruction pointer, thereby an operating system program can perform control functions."

Lee, ¶ 0025. *See also* Lee, ¶ 0026.

> "The boot image memory 108 is capable of containing a non-volatile memory such as a flash memory to store a compressed boot image data. The boot image data can be obtained by compressing an initial storing state of the main memory 104 as a data format."

Lee, ¶ 0038.

> "The BIOS ROM 106 and the boot image memory 108 are capable of setting and storing an initial state of main memory by a manufacturer or a user. Thus, the CPU 102 can reduce a device drive loading time by reading out the compressed boot image from the boot image memory 108 and loading the boot image after decompressing, when boot image is loaded to the main memory 104."

Lee, ¶ 0040.

> "Continually, at step S164, the compressed boot image is read out. At step S166, the compressed boot image is loaded to the main memory 104 after decompressing. At step S168, an instruction pointer (IP) of the CPU

> 102 is set to a specific region of the main memory 104 being loaded the boot image. And then at step S170, the operating system is executed by reading out the boot image from the specific region."

Lee, ¶ 0045. *See also* Lee, ¶ 0048.

> "Continually, at step S224, a compressed boot image 216 is loaded from the CD-ROM 214, and the boot image is loaded to the main memory 206 after decompressing in step S226. At step S228, an instruction pointer 204 of the CPU 202 is set to a specific region 208 of the main memory 206 being loaded the boot image. At step S230, the operating system is executed by reading out the boot image from the specific region 208 of the main memory 206. As a result, the operating system can perform control functions."

Lee, ¶ 0051.

> "The CPU 302 reads out the boot image 324 from the hard disk drive 320 and loads it to the main memory 304, under control of the BIOS. In other words, the CPU 302 decompresses the compressed boot image from the specific region of the hard disk drive 320, and loads it to a specific region of the main memory 304. The CPU 302 reads out the location information of the boot image from the BIOS ROM 306, and then reads out the boot image from the specific region of the main memory 304 according to the information. Thus, the operating system can perform control functions 15 by setting the IP of the CPU 302 to the specific region of the main memory 304."

Lee, ¶ 0054.

Appendix C12
Invalidity of U.S. Patent 8,880,862 based on Lee

1.2 accessing the loaded portion of the boot data in the compressed form from memory;	Lee, as evidenced by the example citations below, discloses "accessing the loaded portion of the boot data in the compressed form from memory;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded portion of the boot data in the compressed form from memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lee discloses this limitation:

> "The BIOS ROM 106 and the boot image memory 108 are capable of setting and storing an initial state of main memory by a manufacturer or a user. Thus, the CPU 102 can reduce a device drive loading time by reading out the compressed boot image from the boot image memory 108 and loading the boot image after decompressing, when boot image is loaded to the main memory 104."

Lee, ¶ 0040.

> "Continually, at step S164, the compressed boot image is read out. At step S166, the compressed boot image is loaded to the main memory 104 after decompressing. At step S168, an instruction pointer (IP) of the CPU 102 is set to a specific region of the main memory 104 being loaded the boot image. And then at step S170, the operating system is executed by reading out the boot image from the specific region."

Lee, ¶ 0045.

> "Continually, at step S224, a compressed boot image 216 is loaded from the CD-ROM 214, and the boot image is loaded to the main memory 206 after decompressing in step S226. At step S228, an instruction pointer 204 of the CPU 202 is set to a specific region 208 of the main memory 206 being loaded the boot image. At step S230, the operating system is executed by reading out the boot image from the specific region 208 of the main memory 206. As a result, the operating system can perform control functions."

Lee, ¶ 0051.

> "The CPU 302 reads out the boot image 324 from the hard disk drive 320

> and loads it to the main memory 304, under control of the BIOS. In other words, the CPU 302 decompresses the compressed boot image from the specific region of the hard disk drive 320, and loads it to a specific region of the main memory 304. The CPU 302 reads out the location information of the boot image from the BIOS ROM 306, and then reads out the boot image from the specific region of the main memory 304 according to the information. Thus, the operating system can perform control functions 15 by setting the IP of the CPU 302 to the specific region of the main memory 304."

Lee, ¶ 0054.

1.3 decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form; and	Lee, as evidenced by the example citations below, discloses "decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lee discloses this limitation:

> "The BIOS ROM 106 and the boot image memory 108 are capable of setting and storing an initial state of main memory by a manufacturer or a user. Thus, the CPU 102 can reduce a device drive loading time by reading out the compressed boot image from the boot image memory 108 and loading the boot image after decompressing, when boot image is loaded to the main memory 104."

Lee, ¶ 0040.

> "Continually, at step S164, the compressed boot image is read out. At step S166, the compressed boot image is loaded to the main memory 104 after decompressing. At step S168, an instruction pointer (IP) of the CPU 102 is set to a specific region of the main memory 104 being loaded the boot image. And then at step S170, the operating system is executed by reading out the boot image from the specific region."

Lee, ¶ 0045.

> "Continually, at step S224, a compressed boot image 216 is loaded from the CD-ROM 214, and the boot image is loaded to the main memory 206 after decompressing in step S226. At step S228, an instruction pointer 204 of the CPU 202 is set to a specific region 208 of the main memory 206 being loaded the boot image. At step S230, the operating system is executed by reading out the boot image from the specific region 208 of the main memory 206. As a result, the operating system can perform

> control functions."

Lee, ¶ 0051.

> "The CPU 302 reads out the boot image 324 from the hard disk drive 320 and loads it to the main memory 304, under control of the BIOS. In other words, the CPU 302 decompresses the compressed boot image from the specific region of the hard disk drive 320, and loads it to a specific region of the main memory 304. The CPU 302 reads out the location information of the boot image from the BIOS ROM 306, and then reads out the boot image from the specific region of the main memory 304 according to the information. Thus, the operating system can perform control functions 15 by setting the IP of the CPU 302 to the specific region of the main memory 304."

Lee, ¶ 0054.

1.4.1 updating the boot data list,	Lee, as evidenced by the example citations below, discloses “updating the boot data list,”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.	

1.4.2 wherein the decompressed portion of boot data comprises a portion of the operating system.	Lee, as evidenced by the example citations below, discloses "wherein the decompressed portion of boot data comprises a portion of the operating system."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the decompressed portion of boot data comprises a portion of the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lee discloses this limitation:

> "The BIOS ROM 106 and the boot image memory 108 are capable of setting and storing an initial state of main memory by a manufacturer or a user. Thus, the CPU 102 can reduce a device drive loading time by reading out the compressed boot image from the boot image memory 108 and loading the boot image after decompressing, when boot image is loaded to the main memory 104."

Lee, ¶ 0040.

> "The CPU 302 reads out the boot image 324 from the hard disk drive 320 and loads it to the main memory 304, under control of the BIOS. In other words, the CPU 302 decompresses the compressed boot image from the specific region of the hard disk drive 320, and loads it to a specific region of the main memory 304. The CPU 302 reads out the location information of the boot image from the BIOS ROM 306, and then reads out the boot image from the specific region of the main memory 304 according to the information. Thus, the operating system can perform control functions 15 by setting the IP of the CPU 302 to the specific region of the main memory 304."

Lee, ¶ 0054.

2. The method of claim 1, wherein the updating comprises: associating additional boot data with the boot data list.	Lee, as evidenced by the example citations below, discloses "wherein the updating comprises: associating additional boot data with the boot data list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein associating additional boot data with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1.4.1 above.	

3. The method of claim 1, wherein the updating comprises: removing an association of additional boot data that is associated with the boot data list from the boot data list.	Lee, as evidenced by the example citations below, discloses "wherein the updating comprises: removing an association of additional boot data that is associated with the boot data list from the boot data list."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein removing an association of additional boot data that is associated with the boot data list from the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1.4.1 above.	

4. The method of claim 1, wherein the updating comprises: associating additional boot data with the boot data list; and compressing a portion of the additional boot data.	Lee, as evidenced by the example citations below, discloses "wherein the updating comprises: associating additional boot data with the boot data list; and compressing a portion of the additional boot data."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein associating additional boot data with the boot data list; and compressing a portion of the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 1.4.1, and 2 above.	

5 (Preamble) A method for booting a computer system, the method comprising:	Lee, as evidenced by the example citations below, discloses "a method for booting a computer system, the method comprising:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 1-1.4.2 above. *See also* **Add disclosure from '608 Patent Claims 1.1 and 1.2.** "The ROM-based code attempts to establish communication with a so-called "boot device". A boot device holds information that is necessary to boot the system. In attempting to establish communication with a boot device, the ROM-based code operates according to a so-called "boot order."" Lee, ¶ 0012. "More particularly, the ROM-based code attempts to retrieve a so-called "master boot record" from a particular sector of the diskette. If the communication attempt is successful, the ROM-based code uses that device as the boot device. If not, the ROM-based code proceeds to attempt communication With a device of the next boot order, e.g., local media." Lee, ¶ 0012. "In order to attain the above objects, according to an aspect of the present invention, there is provided a computer system having a central processing unit; a main and/or auxiliary power supply for supplying main and/or auxiliary power of the computer system; a boot image storing device for storing a boot image of the computer system; a main memory for storing the boot image from the boot image storing device by receiving the auxiliary power when the main power is shut off; and a composition memory for setting an instruction pointer of the central processing unit to a specific region of the main memory storing the boot image; wherein the central processing unit loads the boot image from the	

> specific region of the main memory in response to the instruction pointer, thereby an operating system program can perform control functions."

Lee, ¶ 0025. *See also* Lee, ¶ 0026.

> "The boot image memory 108 is capable of containing a non-volatile memory such as a flash memory to store a compressed boot image data. The boot image data can be obtained by compressing an initial storing state of the main memory 104 as a data format."

Lee, ¶ 0038.

> "The BIOS ROM 106 and the boot image memory 108 are capable of setting and storing an initial state of main memory by a manufacturer or a user. Thus, the CPU 102 can reduce a device drive loading time by reading out the compressed boot image from the boot image memory 108 and loading the boot image after decompressing, when boot image is loaded to the main memory 104."

Lee, ¶ 0040.

> "FIG. 5 is a flow chart for illustrating a method for booting the computer system 200 shown in FIG. 4. The control flow is a program stored in the BIOS ROM 210, and is executed by the CPU 202 according to processing steps of the BIOS."

Lee, ¶ 0050. *See also* Lee, ¶ 0053.

> "When power is applied to a computer system, a portion of the computer system typically called the "initialization hardware" electronically detects the "power-on" condition and, in response to such a detection, forces certain circuitry of the system to a known state. For example, the CPU typically includes an instruction pointer (IP), which holds a memory address from which the CPU fetches an instruction to be executed by the CPU. The initialization hardware typically electronically forces the IP to an initial address so that the CPU may begin fetching and executing instructions from this initial address. The ROM is prerecorded With computer instructions, referred to below as the "ROM-based code." As a result, shortly after power on, the CPU begins executing the ROM-based code."

Lee, ¶ 0011.

> "The BIOS ROM 106 controls POST routine, interrupt processing, and

> system environment setting, according to initializing steps of the computer system 100. Especially, the BIOS ROM 106 sets the instruction pointer (IP)."

Lee, ¶ 0039.

> "In other words, at the step S142, the CPU 102 executes the operating system if a successful boot is detected through a POST routine. Therefore, a certain application program can be executed in the operating system environment."

Lee, ¶ 0042.

> "Referring to FIG. 3, the computer system 100 is powered on in step S160, and POST routine is performed in step S162."

Lee, ¶ 0045. *See also* Lee, ¶ 0051.

5.1 storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory;	Lee, as evidenced by the example citations below, discloses "storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1.1 above. *See also* ""At step S148, the generated boot image is stored to the boot image memory 108 after compressing, and then the computer system 100 is rebooted." Lee, ¶ 0043.	

5.2 loading the stored compressed boot data from the first memory;	Lee, as evidenced by the example citations below, discloses "loading the stored compressed boot data from the first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the stored compressed boot data from the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1.1 above.	

5.3 accessing the loaded compressed boot data;	Lee, as evidenced by the example citations below, discloses "accessing the loaded compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1.2 above.	

5.4 decompressing the accessed compressed boot data;	Lee, as evidenced by the example citations below, discloses "decompressing the accessed compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1.3 above.	

5.5 utilizing the decompressed boot data to at least partially boot the computer system;	Lee, as evidenced by the example citations below, discloses "utilizing the decompressed boot data to at least partially boot the computer system"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed boot data to at least partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lee discloses this limitation:

See Claims 1.3 and 1.4.2 above.

See also

> "The BIOS ROM 106 and the boot image memory 108 are capable of setting and storing an initial state of main memory by a manufacturer or a user. Thus, the CPU 102 can reduce a device drive loading time by reading out the compressed boot image from the boot image memory 108 and loading the boot image after decompressing, when boot image is loaded to the main memory 104."

Lee, ¶ 0040.

> "Continually, at step S164, the compressed boot image is read out. At step S166, the compressed boot image is loaded to the main memory 104 after decompressing. At step S168, an instruction pointer (IP) of the CPU 102 is set to a specific region of the main memory 104 being loaded the boot image. And then at step S170, the operating system is executed by reading out the boot image from the specific region."

Lee, ¶ 0045. *See also* Lee, ¶ 0048.

> "Continually, at step S224, a compressed boot image 216 is loaded from the CD-ROM 214, and the boot image is loaded to the main memory 206 after decompressing in step S226. At step S228, an instruction pointer 204 of the CPU 202 is set to a specific region 208 of the main memory 206 being loaded the boot image. At step S230, the operating system is executed by reading out the boot image from the specific region 208 of the main memory 206. As a result, the operating system can perform control functions."

Lee, ¶ 0051.

> "The CPU 302 reads out the boot image 324 from the hard disk drive 320 and loads it to the main memory 304, under control of the BIOS. In other words, the CPU 302 decompresses the compressed boot image from the specific region of the hard disk drive 320, and loads it to a specific region of the main memory 304. The CPU 302 reads out the location information of the boot image from the BIOS ROM 306, and then reads out the boot image from the specific region of the main memory 304 according to the information. Thus, the operating system can perform control functions 15 by setting the IP of the CPU 302 to the specific region of the main memory 304."

Lee, ¶ 0054.

5.6 updating the boot data list;	Lee, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1.4.1 above.	

5.7 wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form.	Lee, as evidenced by the example citations below, discloses "wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 1-1.3 above.	

6 (Preamble) A system comprising: a processor;	Lee, as evidenced by the example citations below, discloses "a system comprising: a processor:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system comprising: a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lee discloses this limitation:

> "When power is applied to a computer system, a portion of the computer system typically called the "initialization hardware" electronically detects the "power-on" condition and, in response to such a detection, forces certain circuitry of the system to a known state. For example, the CPU typically includes an instruction pointer (IP), which holds a memory address from which the CPU fetches an instruction to be executed by the CPU. The initialization hardware typically electronically forces the IP to an initial address so that the CPU may begin fetching and executing instructions from this initial address. The ROM is prerecorded With computer instructions, referred to below as the "ROM-based code." As a result, shortly after power on, the CPU begins executing the ROM-based code."

Lee, ¶ 0011.

> "The BIOS ROM 106 controls POST routine, interrupt processing, and system environment setting, according to initializing steps of the computer system 100. Especially, the BIOS ROM 106 sets the instruction pointer (IP)."

Lee, ¶ 0039.

> "In other words, at the step S142, the CPU 102 executes the operating system if a successful boot is detected through a POST routine. Therefore, a certain application program can be executed in the operating system environment."

Lee, ¶ 0042.

> "Referring to FIG. 3, the computer system 100 is powered on in step S160, and POST routine is performed in step S162."

Lee, ¶ 0045. *See also* Lee, ¶ 0051.

6.1 a memory; and	Lee, as evidenced by the example citations below, discloses “a memory”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a memory.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Lee discloses this limitation: *See* Claims 1.1 and 1.2 above.	

6.2 a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor	Lee, as evidenced by the example citations below, discloses “a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Lee discloses this limitation: *See* Claims 1.1 and 1.2 above. *See also* > “When power is applied to a computer system, a portion of the computer system typically called the “initialization hardware” electronically detects the “power-on” condition and, in response to such a detection, forces certain circuitry of the system to a known state. For example, the CPU typically includes an instruction pointer (IP), which holds a memory address from which the CPU fetches an instruction to be executed by the CPU. The initialization hardware typically electronically forces the IP to an initial address so that the CPU may begin fetching and executing instructions from this initial address. The ROM is prerecorded With computer instructions, referred to below as the “ROM-based code.” As a result, shortly after power on, the CPU begins executing the ROM-based code.” Lee, ¶ 0011. > “The BIOS ROM 106 controls POST routine, interrupt processing, and system environment setting, according to initializing steps of the computer system 100. Especially, the BIOS ROM 106 sets the instruction pointer (IP).” Lee, ¶ 0039. > “In other words, at the step S142, the CPU 102 executes the operating system if a successful boot is detected through a POST routine. Therefore, a certain application program can be executed in the operating system	

> environment."

Lee, ¶ 0042.

> "Referring to FIG. 3, the computer system 100 is powered on in step S160, and POST routine is performed in step S162."

Lee, ¶ 0045. *See also* Lee, ¶ 0051.

6.3 wherein the processor is configured:	Lee, as evidenced by the example citations below, discloses "wherein the processor is configured:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the processor is configured), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lee discloses this limitation:

> "When power is applied to a computer system, a portion of the computer system typically called the "initialization hardware" electronically detects the "power-on" condition and, in response to such a detection, forces certain circuitry of the system to a known state. For example, the CPU typically includes an instruction pointer (IP), which holds a memory address from which the CPU fetches an instruction to be executed by the CPU. The initialization hardware typically electronically forces the IP to an initial address so that the CPU may begin fetching and executing instructions from this initial address. The ROM is prerecorded With computer instructions, referred to below as the "ROM-based code." As a result, shortly after power on, the CPU begins executing the ROM-based code."

Lee, ¶ 0011.

> "The BIOS ROM 106 controls POST routine, interrupt processing, and system environment setting, according to initializing steps of the computer system 100. Especially, the BIOS ROM 106 sets the instruction pointer (IP)."

Lee, ¶ 0039.

> "In other words, at the step S142, the CPU 102 executes the operating system if a successful boot is detected through a POST routine. Therefore, a certain application program can be executed in the operating system environment."

Lee, ¶ 0042.

> "Referring to FIG. 3, the computer system 100 is powered on in step S160, and POST routine is performed in step S162."

Lee, ¶ 0045. *See also* Lee, ¶ 0051.

6.4 to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory,	Lee, as evidenced by the example citations below, discloses "to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1.1 above.	

6.5 to access the loaded portion of the boot data in the compressed form,	Lee, as evidenced by the example citations below, discloses "to access the loaded portion of the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to access the loaded portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1.2 above.	

6.6 to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data, and	Lee, as evidenced by the example citations below, discloses "to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lee discloses this limitation:

See Claim 1.3 above.

6.7 to update the boot data list.	Lee, as evidenced by the example citations below, discloses "to update the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to update the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1.4.1 above.	

Appendix C12
Invalidity of U.S. Patent 8,880,862 based on Lee

8 (Preamble) A method of loading an operating system for booting a computer system, comprising:	Lee, as evidenced by the example citations below, discloses "A method of loading an operating system for booting a computer system, comprising:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method of loading an operating system for booting a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1 (Preamble) above.	

8.1 storing a portion of the operating system in a compressed form in a first memory;	Lee, as evidenced by the example citations below, discloses "storing a portion of the operating system in a compressed form in a first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing a portion of the operating system in a compressed form in a first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 6.2 above.	

8.2 loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list;	Lee, as evidenced by the example citations below, discloses "loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1.1 above.	

8.3 accessing the loaded portion of the operating system from the second memory in the compressed form;	Lee, as evidenced by the example citations below, discloses "accessing the loaded portion of the operating system from the second memory in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded portion of the operating system from the second memory in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1.2 above.	

8.4 decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system;	Lee, as evidenced by the example citations below, discloses "decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 1.3 and 1.4.2 above.	

8.5 utilizing the decompressed portion of the operating system to at least partially boot the computer system; and	Lee, as evidenced by the example citations below, discloses "utilizing the decompressed portion of the operating system to at least partially boot the computer system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed portion of the operating system to at least partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 1.3 and 1.4.2 above. *See also* "The BIOS ROM 106 and the boot image memory 108 are capable of setting and storing an initial state of main memory by a manufacturer or a user. Thus, the CPU 102 can reduce a device drive loading time by reading out the compressed boot image from the boot image memory 108 and loading the boot image after decompressing, when boot image is loaded to the main memory 104." Lee, ¶ 0040. "Continually, at step S164, the compressed boot image is read out. At step S166, the compressed boot image is loaded to the main memory 104 after decompressing. At step S168, an instruction pointer (IP) of the CPU 102 is set to a specific region of the main memory 104 being loaded the boot image. And then at step S170, the operating system is executed by reading out the boot image from the specific region." Lee, ¶ 0045. *See also* Lee, ¶ 0048. "Continually, at step S224, a compressed boot image 216 is loaded from the CD-ROM 214, and the boot image is loaded to the main memory 206 after decompressing in step S226. At step S228, an instruction pointer 204 of the CPU 202 is set to a specific region 208 of the main memory 206 being loaded the boot image. At step S230, the operating system is executed by reading out the boot image from the specific region 208 of the main memory 206. As a result, the operating system can perform	

> control functions."

Lee, ¶ 0051.

> "The CPU 302 reads out the boot image 324 from the hard disk drive 320 and loads it to the main memory 304, under control of the BIOS. In other words, the CPU 302 decompresses the compressed boot image from the specific region of the hard disk drive 320, and loads it to a specific region of the main memory 304. The CPU 302 reads out the location information of the boot image from the BIOS ROM 306, and then reads out the boot image from the specific region of the main memory 304 according to the information. Thus, the operating system can perform control functions 15 by setting the IP of the CPU 302 to the specific region of the main memory 304."

Lee, ¶ 0054.

8.6 updating the boot data list,	Lee, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1.4.1 above.	

8.7 wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form.	Lee, as evidenced by the example citations below, discloses "wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 1-1.3 and 5.7 above.	

9.1 The method of claim 8, further comprising: compressing an additional portion of the operating system that is not associated with the boot data list; and	Lee, as evidenced by the example citations below, discloses "the method of claim 8, further comprising: compressing an additional portion of the operating system that is not associated with the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, compressing an additional portion of the operating system that is not associated with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1.1 above.	

9.2 storing the additional portion of the operating system in the first memory, and	Lee, as evidenced by the example citations below, discloses "storing the additional portion of the operating system in the first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing the additional portion of the operating system in the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 8.1 above.	

9.3 wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system.	Lee, as evidenced by the example citations below, discloses “wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the utilizing comprises: utilizing the stored additional portion of the operating system to at least further partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Lee discloses this limitation: *See* Claim 8.5 above.	

10. The method of claim 9, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder.	Lee, as evidenced by the example citations below, discloses "the method of claim 9, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the compressing comprises: compressing the additional portion of the operating system with a data compression encoder), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee disclose this limitation: *See* Claim 9.1 above. *See also* "At step S148, the generated boot image is stored to the boot image memory 108 after compressing, and then the computer system 100 is rebooted." Lee, ¶ 0043.	

11 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	Lee, as evidenced by the example citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, comprising:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1 (Preamble) above.	

11.1 loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system;	Lee, as evidenced by the example citations below, discloses "loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading boot data in a compressed form that is associated with a boot data list from a boot device into a memory upon initialization of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lee discloses this limitation:

See Claim 1.1 above.

See also

> "When power is applied to a computer system, a portion of the computer system typically called the "initialization hardware" electronically detects the "power-on" condition and, in response to such a detection, forces certain circuitry of the system to a known state. For example, the CPU typically includes an instruction pointer (IP), which holds a memory address from which the CPU fetches an instruction to be executed by the CPU. The initialization hardware typically electronically forces the IP to an initial address so that the CPU may begin fetching and executing instructions from this initial address. The ROM is prerecorded With computer instructions, referred to below as the "ROM-based code." As a result, shortly after power on, the CPU begins executing the ROM-based code."

Lee, ¶ 0011.

> "The BIOS ROM 106 controls POST routine, interrupt processing, and system environment setting, according to initializing steps of the computer system 100. Especially, the BIOS ROM 106 sets the instruction pointer (IP)."

Lee, ¶ 0039.

> "In other words, at the step S142, the CPU 102 executes the operating system if a successful boot is detected through a POST routine. Therefore,

> a certain application program can be executed in the operating system environment."

Lee, ¶ 0042.

> "Referring to FIG. 3, the computer system 100 is powered on in step S160, and POST routine is performed in step S162."

Lee, ¶ 0045. *See also* Lee, ¶ 0051.

11.2 accessing the loaded boot data in compressed form from the memory;	Lee, as evidenced by the example citations below, discloses "accessing the loaded boot data in compressed form from the memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded boot data in compressed form from the memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1.2 above.	

11.3 decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form;	Lee, as evidenced by the example citations below, discloses "decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed boot data in compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1.3 above.	

11.4 utilizing the decompressed boot data to load at least a portion of the operating system for the computer system; and	Lee, as evidenced by the example citations below, discloses "utilizing the decompressed boot data to load at least a portion of the operating system for the computer system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed boot data to load at least a portion of the operating system for the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: "The BIOS ROM 106 and the boot image memory 108 are capable of setting and storing an initial state of main memory by a manufacturer or a user. Thus, the CPU 102 can reduce a device drive loading time by reading out the compressed boot image from the boot image memory 108 and loading the boot image after decompressing, when boot image is loaded to the main memory 104." Lee, ¶ 0040. "Continually, at step S164, the compressed boot image is read out. At step S166, the compressed boot image is loaded to the main memory 104 after decompressing. At step S168, an instruction pointer (IP) of the CPU 102 is set to a specific region of the main memory 104 being loaded the boot image. And then at step S170, the operating system is executed by reading out the boot image from the specific region." Lee, ¶ 0045. *See also* Lee, ¶ 0048. "Continually, at step S224, a compressed boot image 216 is loaded from the CD-ROM 214, and the boot image is loaded to the main memory 206 after decompressing in step S226. At step S228, an instruction pointer 204 of the CPU 202 is set to a specific region 208 of the main memory 206 being loaded the boot image. At step S230, the operating system is executed by reading out the boot image from the specific region 208 of the main memory 206. As a result, the operating system can perform control functions." Lee, ¶ 0051. "The CPU 302 reads out the boot image 324 from the hard disk drive 320	

and loads it to the main memory 304, under control of the BIOS. In other words, the CPU 302 decompresses the compressed boot image from the specific region of the hard disk drive 320, and loads it to a specific region of the main memory 304. The CPU 302 reads out the location information of the boot image from the BIOS ROM 306, and then reads out the boot image from the specific region of the main memory 304 according to the information. Thus, the operating system can perform control functions 15 by setting the IP of the CPU 302 to the specific region of the main memory 304."

Lee, ¶ 0054.

11.5 updating the boot data list.	Lee, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1.4.1 above.	

13 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	Lee, as evidenced by the example citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, comprising"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1 (Preamble) above.	

Appendix C12
Invalidity of U.S. Patent 8,880,862 based on Lee

13.1 loading boot data in a compressed form that is associated with a boot data list from a boot device;	Lee, as evidenced by the example citations below, discloses "loading boot data in a compressed form that is associated with a boot data list from a boot device"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading boot data in a compressed form that is associated with a boot data list from a boot device), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1.1 above.	

13.2 accessing the loaded boot data in the compressed form;	Lee, as evidenced by the example citations below, discloses "accessing the loaded boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1.2 above.	

13.3 decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form;	Lee, as evidenced by the example citations below, discloses "decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed boot data in the compressed form at a rate that decreases a time to load the operating system relative to loading the operating system with the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1.3 above.	

13.4 updating the boot data list.	Lee, as evidenced by the example citations below, discloses “updating the boot data list”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Lee discloses this limitation: *See* Claim 1.4.1 above.	

Appendix C12
Invalidity of U.S. Patent 8,880,862 based on Lee

14 (Preamble) A method for providing accelerated loading of an operating system in a computer system, comprising:	Lee, as evidenced by the example citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, comprising"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method for providing accelerated loading of an operating system in a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1 (Preamble) above.	

14.1 accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list;	Lee, as evidenced by the example citations below, discloses "accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing boot data for booting the computer system, wherein a portion of the boot data is in a compressed form and is associated with a boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lee discloses this limitation:

See Claims 1.1 and 1.2 above.

See also

> "The ROM-based code attempts to establish communication with a so-called "boot device". A boot device holds information that is necessary to boot the system. In attempting to establish communication with a boot device, the ROM-based code operates according to a so-called "boot order.""

Lee, ¶ 0012.

> "More particularly, the ROM-based code attempts to retrieve a so-called "master boot record" from a particular sector of the diskette. If the communication attempt is successful, the ROM-based code uses that device as the boot device. If not, the ROM-based code proceeds to attempt communication With a device of the next boot order, e.g., local media."

Lee, ¶ 0012.

> "In order to attain the above objects, according to an aspect of the present invention, there is provided a computer system having a central processing unit; a main and/or auxiliary power supply for supplying main and/or auxiliary power of the computer system; a boot image storing device for storing a boot image of the computer system; a main memory for storing the boot image from the boot image storing device by receiving the auxiliary power when the main power is shut off; and a composition memory for setting an instruction pointer of the central

> processing unit to a specific region of the main memory storing the boot image; wherein the central processing unit loads the boot image from the specific region of the main memory in response to the instruction pointer, thereby an operating system program can perform control functions."

Lee, ¶ 0025. *See also* Lee, ¶ 0026.

> "The boot image memory 108 is capable of containing a non-volatile memory such as a flash memory to store a compressed boot image data. The boot image data can be obtained by compressing an initial storing state of the main memory 104 as a data format."

Lee, ¶ 0038.

> "The BIOS ROM 106 and the boot image memory 108 are capable of setting and storing an initial state of main memory by a manufacturer or a user. Thus, the CPU 102 can reduce a device drive loading time by reading out the compressed boot image from the boot image memory 108 and loading the boot image after decompressing, when boot image is loaded to the main memory 104."

Lee, ¶ 0040.

> "FIG. 5 is a flow chart for illustrating a method for booting the computer system 200 shown in FIG. 4. The control flow is a program stored in the BIOS ROM 210, and is executed by the CPU 202 according to processing steps of the BIOS."

Lee, ¶ 0050. *See also* Lee, ¶ 0053.

14.2 loading the boot data into a memory; and	Lee, as evidenced by the example citations below, discloses "loading the boot data into a memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the boot data into a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1.1 above.	

14.3 servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form; and	Lee, as evidenced by the example citations below, discloses "servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, servicing a request for the boot data from the computer system to access the loaded compressed boot data and to decompress the accessed compressed boot data at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing the boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lee discloses this limitation:

See Claims 1.2 and 1.3 above.

See also

> "The BIOS ROM 106 and the boot image memory 108 are capable of setting and storing an initial state of main memory by a manufacturer or a user. Thus, the CPU 102 can reduce a device drive loading time by reading out the compressed boot image from the boot image memory 108 and loading the boot image after decompressing, when boot image is loaded to the main memory 104."

Lee, ¶ 0040.

> "Continually, at step S164, the compressed boot image is read out. At step S166, the compressed boot image is loaded to the main memory 104 after decompressing. At step S168, an instruction pointer (IP) of the CPU 102 is set to a specific region of the main memory 104 being loaded the boot image. And then at step S170, the operating system is executed by reading out the boot image from the specific region."

Lee, ¶ 0045. *See also* Lee, ¶ 0048.

> "Continually, at step S224, a compressed boot image 216 is loaded from the CD-ROM 214, and the boot image is loaded to the main memory 206 after decompressing in step S226. At step S228, an instruction pointer 204 of the CPU 202 is set to a specific region 208 of the main memory 206 being loaded the boot image. At step S230, the operating system is executed by reading out the boot image from the specific region 208 of the main memory 206. As a result, the operating system can perform control functions."

Lee, ¶ 0051.

> "The CPU 302 reads out the boot image 324 from the hard disk drive 320 and loads it to the main memory 304, under control of the BIOS. In other words, the CPU 302 decompresses the compressed boot image from the specific region of the hard disk drive 320, and loads it to a specific region of the main memory 304. The CPU 302 reads out the location information of the boot image from the BIOS ROM 306, and then reads out the boot image from the specific region of the main memory 304 according to the information. Thus, the operating system can perform control functions 15 by setting the IP of the CPU 302 to the specific region of the main memory 304."

Lee, ¶ 0054.

14.4 updating the boot data list.	Lee, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 1.4.1 above.	

15. The method of claim 14, wherein the boot data comprises: a program code associated with the operating system.	Lee, as evidenced by the example citations below, discloses "the method of claim 14, wherein the boot data comprises: a program code associated with the operating system."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the boot data comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lee discloses this limitation:

See Claims 1 (Preamble) and 1.1 above.

See also

> "The master boot record, among other things, typically includes information about the particular boot media, such as partitioning information for that diskette, and includes a pointer and an offset to a so-called operating system loader ("OS loader")."

Lee, ¶ 0014. *See also* Lee, ¶ 0026.

16. The method of claim 14, wherein the operating system comprises: a plurality of files.	Lee, as evidenced by the example citations below, discloses "the method of claim 14, wherein the operating system comprises: a plurality of files."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lee discloses this limitation:

> "The computer system is generally controlled and coordinated by an operating system (OS) program. Operating systems, for example Windows 95, Windows 98, Windows NT, Windows 2000, and Windows Millennium Edition of Microsoft Corporation, provide resource management throughout a computer system, including such tasks as process execution and scheduling, memory management, file system services, networking and I/O services, and user interface presentation."

Lee, ¶ 0006.

> "After performing preliminary functions, the OS loader copies into RAM a portion of the operating system known as the "kernel.""

Lee, ¶ 0015.

17. The method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program.	Lee, as evidenced by the example citations below, discloses "the method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the boot data comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lee discloses this limitation:

See Claim 15 above.

> "The initial storing state is capable of executing a certain application program in an operating system program environment."

Lee, ¶ 0038.

> "In other words, at the step S142, the CPU 102 executes the operating system if a successful boot is detected through a POST routine. Therefore, a certain application program can be executed in the operating system environment."

Lee, ¶ 0042.

19.1 The method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:	Lee, as evidenced by the example citations below, discloses “the method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 14, wherein the request for the boot data comprises: a request to access boot data that is not associated with the boot data list, and wherein the updating comprises:), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.	

19.2 associating the accessed boot data that is not associated with the boot data list to the boot data list.	Lee, as evidenced by the example citations below, discloses "associating the accessed boot data that is not associated with the boot data list to the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, associating the accessed boot data that is not associated with the boot data list to the boot data list.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 2 above.	

23. The method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files.	Lee, as evidenced by the example citations below, discloses "the method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the portion of the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 16 above.	

24. The method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system.	Lee, as evidenced by the example citations below, discloses "the method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system.
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the portion of the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 15 above.	

27. The method of claim 1, wherein the memory comprises: a physical memory.	Lee, as evidenced by the example citations below, discloses "the method of claim 1, wherein the memory comprises: a physical memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 1.1 and 1.2 above.	

28. The method of claim 1, wherein the operating system comprises: a plurality of files.	Lee, as evidenced by the example citations below, discloses "the method of claim 1, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 16 above.	

29. The method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program.	Lee, as evidenced by the example citations below, discloses "the method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the boot data comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 15 and 17 above.	

31. The method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access.	Lee, as evidenced by the example citations below, discloses "the method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein the accessing comprises: accessing the loaded portion of the boot data in the compressed form via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 1.1 and 1.2 above.	

32. The method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form.	Lee, as evidenced by the example citations below, discloses "the method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein a form of dictionary encoding was utilized to encode the portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

33. The method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form.	Lee, as evidenced by the example citations below, discloses "the method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 1, wherein Lempel-Ziv encoding was utilized to encode the portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.	

35. The method of claim 5, wherein the compressed boot data represents a plurality of files.	Lee, as evidenced by the example citations below, discloses "the method of claim 5, wherein the compressed boot data represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the compressed boot data represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 16 and 23 above.	

36. The method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system.	Lee, as evidenced by the example citations below, discloses "the method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the compressed boot data comprises: a program code associated with an operating system of the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 15 and 17 above.	

39. The method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory.	Lee, as evidenced by the example citations below, discloses "the method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the loading comprises: loading the stored compressed boot data from the first memory to a second memory, and wherein the second memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 1.1 and 1.2 above.	

40. The method of claim 36, wherein the operating system comprises: a plurality of files.	Lee, as evidenced by the example citations below, discloses "the method of claim 36, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 36, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 16 and 23 above.	

43. The method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access.	Lee, as evidenced by the example citations below, discloses "the method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the accessing comprises: accessing the loaded compressed boot data via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 31 above.	

44. The method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data.	Lee, as evidenced by the example citations below, discloses "the method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein a form of dictionary encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 32 above.	

45. The method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data.	Lee, as evidenced by the example citations below, discloses "the method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 32 and 33 above.	

47. The system of claim 6, wherein the boot data in the compressed form represents a plurality of files.	Lee, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data in the compressed form represents a plurality of files"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lee discloses this limitation:

See Claims 16 and 23 above.

48. The system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system.	Lee, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 15, 17 and 24 above.	

49. The system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form.	Lee, as evidenced by the example citations below, discloses "the system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, further comprising: an encoder configured to compress the boot data to provide the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 10 above.	

50. The system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form.	Lee, as evidenced by the example citations below, discloses "the system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, further comprising: a decoder configured to decompress the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Lee discloses this limitation:

> "Continually, at step S224, a compressed boot image 216 is loaded from the CD-ROM 214, and the boot image is loaded to the main memory 206 after decompressing in step S226. At step S228, an instruction pointer 204 of the CPU 202 is set to a specific region 208 of the main memory 206 being loaded the boot image. At step S230, the operating system is executed by reading out the boot image from the specific region 208 of the main memory 206. As a result, the operating system can perform control functions."

Lee, ¶ 0051.

> "The CPU 302 reads out the boot image 324 from the hard disk drive 320 and loads it to the main memory 304, under control of the BIOS. In other words, the CPU 302 decompresses the compressed boot image from the specific region of the hard disk drive 320, and loads it to a specific region of the main memory 304. The CPU 302 reads out the location information of the boot image from the BIOS ROM 306, and then reads out the boot image from the specific region of the main memory 304 according to the information. Thus, the operating system can perform control functions 15 by setting the IP of the CPU 302 to the specific region of the main memory 304."

Lee, ¶ 0054.

51. The system of claim 6, wherein the first memory comprises: a physical memory.	Lee, as evidenced by the example citations below, discloses "the system of claim 6, wherein the first memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the first memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 27 and 39 above.	

52. The system of claim 6, wherein the boot data the compressed form comprises: a plurality of files.	Lee, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data the compressed form comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data the compressed form comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 16 and 23 above.	

53. The system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program.	Lee, as evidenced by the example citations below, discloses "the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the system of claim 6, wherein the boot data in the compressed form comprises: a program code associated with an operating system of the system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 15, 17 and 24 above.	

59. The method of claim 8, wherein the operating system in the compressed form represents a plurality of files.	Lee, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 16 and 23 above.	

60. The method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system.	Lee, as evidenced by the example citations below, discloses “the method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form comprises: program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Lee discloses this limitation: *See* Claims 15, 17 and 24 above.	

63. The method of claim 8, wherein the second memory comprises: a physical memory.	Lee, as evidenced by the example citations below, discloses "the method of claim 8, wherein the second memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the second memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 27 and 38 above.	

64. The method of claim 8, wherein the operating system comprises: a plurality of files.	Lee, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 16 and 23 above.	

65. The method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program.	Lee, as evidenced by the example citations below, discloses "the method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the operating system in the compressed form comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 15, 17 and 24 above.	

67. The method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access.	Lee, as evidenced by the example citations below, discloses "the method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 8, wherein the accessing comprises: accessing the loaded first portion from the second memory via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 31 above.	

71. The method of claim 11, wherein the boot data in the compressed form represents a plurality of files.	Lee, as evidenced by the example citations below, discloses "the method of claim 11, wherein the boot data in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 16 and 23 above.	

72. The method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system.	Lee, as evidenced by the example citations below, discloses "the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 15, 17 and 24 above.	

75. The method of claim 11, wherein the memory comprises: a physical memory.	Lee, as evidenced by the example citations below, discloses "the method of claim 11, wherein the memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 27 above.	

76. The method of claim 11, wherein the operating system comprises: a plurality of files.	Lee, as evidenced by the example citations below, discloses "the method of claim 11, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 16 and 23 above.	

77. The method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program.	Lee, as evidenced by the example citations below, discloses "the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the boot data in the compressed form comprises: a program code associated with the operating system and an application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 15, 17 and 24 above.	

79. The method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access.	Lee, as evidenced by the example citations below, discloses "the method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein the accessing comprises: accessing the boot data in the compressed form from the memory via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 31 above.	

80. The method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form.	Lee, as evidenced by the example citations below, discloses "the method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein a form of dictionary encoding was utilized to encode the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 32, 33 and 49 above.	

81. The method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form.	Lee, as evidenced by the example citations below, discloses "the method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 11, wherein Lempel-Ziv encoding was utilized to encode the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 32, 33 and 49 above.	

83. The method of claim 13, wherein the boot data in the compressed form represents a plurality of files.	Lee, as evidenced by the example citations below, discloses "the method of claim 13, wherein the boot data in the compressed form represents a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form represents a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 16 and 23 above.	

84. The method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system.	Lee, as evidenced by the example citations below, discloses "the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 15, 17 and 24 above.	

87. The method of claim 13, wherein the memory comprises: a physical memory.	Lee, as evidenced by the example citations below, discloses "the method of claim 13, wherein the memory comprises: a physical memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the memory comprises: a physical memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 27 above.	

Appendix C12
Invalidity of U.S. Patent 8,880,862 based on Lee

88. The method of claim 13, wherein the operating system comprises: a plurality of files.	Lee, as evidenced by the example citations below, discloses "the method of claim 13, wherein the operating system comprises: a plurality of files"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the operating system comprises: a plurality of files), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 16 and 23 above.	

89. The method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program.	Lee, as evidenced by the example citations below, discloses "the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the boot data in the compressed form comprises: a program code associated with the operating system and application program), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 15, 17 and 24 above.	

91. The method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access.	Lee, as evidenced by the example citations below, discloses "the method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein the accessing comprises: accessing the loaded boot data in the compressed form via direct memory access), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 31 above.	

92. The method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data.	Lee, as evidenced by the example citations below, discloses "the method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein a form of dictionary encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 32, 33 and 49 above.	

93. The method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data.	Lee, as evidenced by the example citations below, discloses "the method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 13, wherein Lempel-Ziv encoding was utilized to encode the compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claim 32, 33, and 49 above.	

97.1 The method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list,	Lee, as evidenced by the example citations below, discloses "the method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, further comprising: accessing additional compressed boot data that is not associated with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 9.1-9.3 and 19 above.	

97.2 and wherein the updating comprises: associating the additional compressed boot data with the boot data list.	Lee, as evidenced by the example citations below, discloses "and wherein the updating comprises: associating the additional compressed boot data with the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, and wherein the updating comprises: associating the additional compressed boot data with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 1.4.1, 2, 4, 5.6, and 8.6 above.	

98. The method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list.	Lee, as evidenced by the example citations below, discloses "the method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 5, wherein the updating comprises: disassociating non accessed boot data from the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 1.4.1, 5.6, and 8.6 above.	

107. The method of claim 2, further comprising: storing the updated boot list in a non-volatile memory.	Lee, as evidenced by the example citations below, discloses "the method of claim 2, further comprising: storing the updated boot list in a non-volatile memory."
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 2, further comprising: storing the updated boot list in a non-volatile memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 1.4.1, 5.6, and 8.6 above.	

108. The method of claim 2, further comprising: compressing at least a portion of the additional boot data.	Lee, as evidenced by the example citations below, discloses "the method of claim 2, further comprising: compressing at least a portion of the additional boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 2, further comprising: compressing at least a portion of the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 1.1, 5, 9.1 and 8 above.	

109. The method of claim 108, further comprising: storing the compressed additional boot data.	Lee, as evidenced by the example citations below, discloses "the method of claim 108, further comprising: storing the compressed additional boot data"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, the method of claim 108, further comprising: storing the compressed additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Lee discloses this limitation: *See* Claims 5.1, 8.1 and 9.1 above.	

Appendix C13
Invalidity of U.S. Patent 8,880,862 based on Makinen

U.S. Patent No. 6,237,080 to Makinen ("Makinen") invalidates claims 1-6, 8-11, 13-17, 19, 23-24, 27-29, 31-33, 35-36, 39-40, 43-45, 47-53, 59-60, 63-65, 67, 71-72, 75-77, 79-81, 83-84, 87-89, 91-93, 97-98, and 107-109 of United States Patent No. 8,880,862 ("the '862 Patent") pursuant to 35 U.S.C. § 102 and/or 35 U.S.C. § 103 either alone or in combination with other prior art references, and/or in combination with the knowledge of a person of ordinary skill.

The analysis provided in this chart may in some instances uses Realtime's proposed (or implied) claim constructions, and Apple reserves all rights to challenge these proposed (or implied) constructions. To the extent any of the charted prior art should fail to disclose an element of any claims of the '862 Patent, Apple reserves the right to rely upon the knowledge of one skilled in the art, or any other disclosed prior art, alone or in combination, whether produced by Apple or by Realtime, to show the element and thereby invalidate those claims. Citations given in the chart below are merely representative of the respective elements and are not meant to be exhaustive.

Appendix C13
Invalidity of U.S. Patent 8,880,862 based on Makinen

1 (Preamble) A method for providing accelerated loading of an operating system in a computer system, the method comprising:	Makinen, as evidenced by the exemplary citations below, discloses "a method for providing accelerated loading of an operating system in a computer system, the method comprising:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accelerated loading of an operating system in a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Makinen discloses this limitation:

> A computer having a reduced instruction computer (RISC) architecture has a RISC central processing unit (CPU)(1) coupled to a RAM memory (3) and to a flash ROM memory (4). A set of compressed operating instructions (6,8), including a subset defining a compression method (8), are stored in the flash ROM (4) together with a set of uncompressed instructions (7) defining a compression algorithm. Upon booting of the computer, the uncompressed instructions (7) are read from the ROM (4) by the CPU (1) which then also reads the compressed instructions (6,8), decompresses them according to the decompression process (7), and writes the decompressed instructions (6′,8′) to the RAM (3). The compressed instructions (6,8) can be dynamically altered by the CPU (1), by generating an altered set of uncompressed instructions, compressing these in accordance with the now decompressed compression method (8′), and writing these to the flash ROM (4).

Makinen, Abstract

> According to a first aspect of the present invention there is provided a method of operating apparatus having a central processing unit (CPU) with a reduced instruction set computer (RISC) architecture, and a read only memory (ROM), the method comprising reading a set of compressed RISC operating instructions from the ROM into the CPU, decompressing the compressed instructions in the CPU, and thereafter operating the apparatus in accordance with the decompressed instructions.

Makinen, 1:63-2:4

> According to a second aspect of the present invention there is provided a method of operating apparatus having a central processing unit (CPU) and a read only memory (ROM), the method comprising reading a set of compressed operating instructions from the ROM into the CPU,

> decompressing the compressed instructions in the CPU, and thereafter operating the apparatus in accordance with the decompressed instructions, the method further comprising generating one or more replacement or additional compressed instructions in the CPU and writing the compressed instruction(s) to the ROM.
>
> The above second aspect of the present invention makes it possible to amend the stored compressed instructions in a dynamic manner. This may, for example, allow a user to configure the computer according to his specific needs.
>
> Preferably, the method comprises the step of reading a set of operating instructions from the ROM into the CPU, which instructions define a program for compressing said replacement or additional instruction(s). More preferably, the instructions defining the compression program form part of said set of compressed operating instructions.

Makinen, 2:18-39

> Preferably, the method of the above first or second aspect of the invention comprises writing the decompressed instruction set to a random access memory (RAM). Thereafter, the decompressed instructions are read from the RAM by the CPU. It is noted that RAM typically offers high access speeds compared to slow (e.g. flash) ROM memory, giving a significant increase in system performance. In this case, increased speed also offers reduced power consumption compared to systems which use slow ROM memory and in which power is consumed even when the system is waiting to access the ROM.

Makinen, 2:40-50

> According to a third aspect of the present invention there is provided apparatus having a central processing unit (CPU) a reduced instruction set computer (RISC) architecture, and a read only memory (ROM), there being stored in the ROM a set of compressed RISC operating instructions, the CPU being arranged in use to read the compressed instructions from the ROM, to decompress these instructions, and subsequently to operate the apparatus in accordance with the decompressed instructions.

Makinen, 2:51-59

> According to a fourth aspect of the present invention there is provided apparatus comprising a central processing unit (CPU), a read only memory (ROM), and a set of compressed operating instructions stored in the ROM, the CPU being arranged in use to read the compressed instructions from the ROM, decompress the compressed instructions,

> and thereafter operate the apparatus in accordance with the decompressed instructions, the apparatus being further arranged in use to compress replacement or additional operating instructions and to write these compressed instructions to the ROM.

Makinen, 2:60-3:3:2

Appendix C13
Invalidity of U.S. Patent 8,880,862 based on Makinen

1.1 loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory;	Makinen, as evidenced by the example citations below, discloses "loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading a portion of boot data in a compressed form that is associated with a portion of a boot data list for booting the computer system into a memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Makinen discloses this limitation:

> A computer having a reduced instruction computer (RISC) architecture has a RISC central processing unit (CPU)(1) coupled to a RAM memory (3) and to a flash ROM memory (4). A set of compressed operating instructions (6,8), including a subset defining a compression method (8), are stored in the flash ROM (4) together with a set of uncompressed instructions (7) defining a compression algorithm. Upon booting of the computer, the uncompressed instructions (7) are read from the ROM (4) by the CPU (1) which then also reads the compressed instructions (6,8), decompresses them according to the decompression process (7), and writes the decompressed instructions (6′,8′) to the RAM (3). The compressed instructions (6,8) can be dynamically altered by the CPU (1), by generating an altered set of uncompressed instructions, compressing these in accordance with the now decompressed compression method (8′), and writing these to the flash ROM (4).

Makinen, Abstract

> At the heart of most modern electronic devices is a microprocessor or central processing unit which operates in accordance with a set of software operating instructions which together form an executable program. The instructions are stored in a digital memory which may be internal to the microprocessor or, as is more usually the case, externally connected to the microprocessor. The set of operating instructions generally define the basic input/output system (BIOS) of the microprocessor together with device drivers, libraries, and user applications.

Makinen, 1:10-19

> According to a first aspect of the present invention there is provided a method of operating apparatus having a central processing unit (CPU) with a reduced instruction set computer (RISC) architecture, and a read only memory (ROM), the method comprising reading a set of compressed RISC operating instructions from the ROM into the CPU, decompressing the compressed instructions in the CPU, and thereafter operating the apparatus in accordance with the decompressed instructions.

Makinen, 1:63-2:4

> According to a second aspect of the present invention there is provided a method of operating apparatus having a central processing unit (CPU) and a read only memory (ROM), the method comprising reading a set of compressed operating instructions from the ROM into the CPU, decompressing the compressed instructions in the CPU, and thereafter operating the apparatus in accordance with the decompressed instructions, the method further comprising generating one or more replacement or additional compressed instructions in the CPU and writing the compressed instruction(s) to the ROM.
>
> The above second aspect of the present invention makes it possible to amend the stored compressed instructions in a dynamic manner. This may, for example, allow a user to configure the computer according to his specific needs.
>
> Preferably, the method comprises the step of reading a set of operating instructions from the ROM into the CPU, which instructions define a program for compressing said replacement or additional instruction(s). More preferably, the instructions defining the compression program form part of said set of compressed operating instructions.

Makinen, 2:18-39

> Preferably, the method of the above first or second aspect of the invention comprises writing the decompressed instruction set to a random access memory (RAM). Thereafter, the decompressed instructions are read from the RAM by the CPU. It is noted that RAM typically offers high access speeds compared to slow (e.g. flash) ROM memory, giving a significant increase in system performance. In this case, increased speed also offers reduced power consumption compared to systems which use slow ROM memory and in which power is consumed even when the system is waiting to access the ROM.

Makinen, 2:40-50

> According to a third aspect of the present invention there is provided apparatus having a central processing unit (CPU) a reduced instruction set computer (RISC) architecture, and a read only memory (ROM), there being stored in the ROM a set of compressed RISC operating instructions, the CPU being arranged in use to read the compressed instructions from the ROM, to decompress these instructions, and subsequently to operate the apparatus in accordance with the decompressed instructions.

Makinen, 2:51-59

> According to a fourth aspect of the present invention there is provided apparatus comprising a central processing unit (CPU), a read only memory (ROM), and a set of compressed operating instructions stored in the ROM, the CPU being arranged in use to read the compressed instructions from the ROM, decompress the compressed instructions, and thereafter operate the apparatus in accordance with the decompressed instructions, the apparatus being further arranged in use to compress replacement or additional operating instructions and to write these compressed instructions to the ROM.

Makinen, 2:60-3:3:2

> The flash ROM 4 is used to store a set of RISC operating instructions which define the basic input/output system (BIOS) of the microprocessor as well as the device drivers, libraries, and user applications. The instructions are in compressed form, having previously been compressed using the Pkzip compression program

Makinen, 3:45-50

> In order to enable the microprocessor 1 to decompress the stored compressed instructions, the flash ROM 4 additionally stores a set of instructions, in uncompressed form, which define the Pkzip decompression program. FIG. 2 shows a memory map of the ROM 4 prior to booting the computer, where a part of the memory space is occupied by the compressed instructions 6 and a part is occupied by the uncompressed Pkzip instructions 7. In FIG. 2, the ROM 4 is shown coupled to the microprocessor 1, as is the RAM 3 which at this stage remains empty.

Makinen, 3:56-65

> Upon booting of the computer, the microprocessor 1 is directed by hardwired logic to read the first instruction of the decompressed instruction set 7, from the ROM 4. Thereafter the microprocessor is

> directed, by the decompressed set, to read and decompress the compressed instruction set 6. The expanded instruction set 6′ is then stored in the RAM 3 as shown in FIG. 3. The last operation that the set of decompression instructions perform is to cause the microprocessor 1 to jump to the first instruction in the decompressed code. Thereafter, the computer is operated in accordance with the decompressed operating instructions 6′.

Makinen, 3:66-4:9

> In order to reduce memory requirements, the corresponding Pkzip instructions may be stored in compressed form 8 as illustrated in FIG. 4. During booting, the compressed Pkzip instructions 8 are read from the flash ROM 4 by the microprocessor 1, decompressed, and stored in the RAM 3 as decompressed instructions 8′ together with the decompressed operating instructions 6′ (FIG. 4). Alternatively, the compressed Pkzip instructions 8 may only be read from the flash ROM 4, and subsequently decompressed, when a specific request is made to alter the compressed instructions 6,8. In either case, the microprocessor 1 employs the decompressed Pkzip method to compress an amended version of the operating instructions 6′,8′ and writes the compressed instructions to the corresponding areas of the flash ROM.

Makinen, 4:18-32

See also Makinen, Figs. 2-4

1.2 accessing the loaded portion of the boot data in the compressed form from memory;	Makinen, as evidenced by the example citations below, discloses "accessing the loaded portion of the boot data in the compressed form from memory;"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded portion of the boot data in the compressed form from memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Makinen discloses this limitation:

> A computer having a reduced instruction computer (RISC) architecture has a RISC central processing unit (CPU)(1) coupled to a RAM memory (3) and to a flash ROM memory (4). A set of compressed operating instructions (6,8), including a subset defining a compression method (8), are stored in the flash ROM (4) together with a set of uncompressed instructions (7) defining a compression algorithm. Upon booting of the computer, the uncompressed instructions (7) are read from the ROM (4) by the CPU (1) which then also reads the compressed instructions (6,8), decompresses them according to the decompression process (7), and writes the decompressed instructions (6′,8′) to the RAM (3). The compressed instructions (6,8) can be dynamically altered by the CPU (1), by generating an altered set of uncompressed instructions, compressing these in accordance with the now decompressed compression method (8′), and writing these to the flash ROM (4).

Makinen, Abstract

> At the heart of most modern electronic devices is a microprocessor or central processing unit which operates in accordance with a set of software operating instructions which together form an executable program. The instructions are stored in a digital memory which may be internal to the microprocessor or, as is more usually the case, externally connected to the microprocessor. The set of operating instructions generally define the basic input/output system (BIOS) of the microprocessor together with device drivers, libraries, and user applications.

Makinen, 1:10-19

> According to a first aspect of the present invention there is provided a method of operating apparatus having a central processing unit (CPU)

with a reduced instruction set computer (RISC) architecture, and a read only memory (ROM), the method comprising reading a set of compressed RISC operating instructions from the ROM into the CPU, decompressing the compressed instructions in the CPU, and thereafter operating the apparatus in accordance with the decompressed instructions.

Makinen, 1:63-2:4

> According to a second aspect of the present invention there is provided a method of operating apparatus having a central processing unit (CPU) and a read only memory (ROM), the method comprising reading a set of compressed operating instructions from the ROM into the CPU, decompressing the compressed instructions in the CPU, and thereafter operating the apparatus in accordance with the decompressed instructions, the method further comprising generating one or more replacement or additional compressed instructions in the CPU and writing the compressed instruction(s) to the ROM.
>
> The above second aspect of the present invention makes it possible to amend the stored compressed instructions in a dynamic manner. This may, for example, allow a user to configure the computer according to his specific needs.
>
> Preferably, the method comprises the step of reading a set of operating instructions from the ROM into the CPU, which instructions define a program for compressing said replacement or additional instruction(s). More preferably, the instructions defining the compression program form part of said set of compressed operating instructions.

Makinen, 2:18-39

> Preferably, the method of the above first or second aspect of the invention comprises writing the decompressed instruction set to a random access memory (RAM). Thereafter, the decompressed instructions are read from the RAM by the CPU. It is noted that RAM typically offers high access speeds compared to slow (e.g. flash) ROM memory, giving a significant increase in system performance. In this case, increased speed also offers reduced power consumption compared to systems which use slow ROM memory and in which power is consumed even when the system is waiting to access the ROM.

Makinen, 2:40-50

> According to a third aspect of the present invention there is provided apparatus having a central processing unit (CPU) a reduced instruction

set computer (RISC) architecture, and a read only memory (ROM), there being stored in the ROM a set of compressed RISC operating instructions, the CPU being arranged in use to read the compressed instructions from the ROM, to decompress these instructions, and subsequently to operate the apparatus in accordance with the decompressed instructions.

Makinen, 2:51-59

> According to a fourth aspect of the present invention there is provided apparatus comprising a central processing unit (CPU), a read only memory (ROM), and a set of compressed operating instructions stored in the ROM, the CPU being arranged in use to read the compressed instructions from the ROM, decompress the compressed instructions, and thereafter operate the apparatus in accordance with the decompressed instructions, the apparatus being further arranged in use to compress replacement or additional operating instructions and to write these compressed instructions to the ROM.

Makinen, 2:60-3:3:2

> The flash ROM 4 is used to store a set of RISC operating instructions which define the basic input/output system (BIOS) of the microprocessor as well as the device drivers, libraries, and user applications. The instructions are in compressed form, having previously been compressed using the Pkzip compression program

Makinen, 3:45-50

> In order to enable the microprocessor 1 to decompress the stored compressed instructions, the flash ROM 4 additionally stores a set of instructions, in uncompressed form, which define the Pkzip decompression program. FIG. 2 shows a memory map of the ROM 4 prior to booting the computer, where a part of the memory space is occupied by the compressed instructions 6 and a part is occupied by the uncompressed Pkzip instructions 7. In FIG. 2, the ROM 4 is shown coupled to the microprocessor 1, as is the RAM 3 which at this stage remains empty.

Makinen, 3:56-65

> Upon booting of the computer, the microprocessor 1 is directed by hardwired logic to read the first instruction of the decompressed instruction set 7, from the ROM 4. Thereafter the microprocessor is directed, by the decompressed set, to read and decompress the compressed instruction set 6. The expanded instruction set 6′ is then

> stored in the RAM 3 as shown in FIG. 3. The last operation that the set of decompression instructions perform is to cause the microprocessor 1 to jump to the first instruction in the decompressed code. Thereafter, the computer is operated in accordance with the decompressed operating instructions 6′.

Makinen, 3:66-4:9

> In order to reduce memory requirements, the corresponding Pkzip instructions may be stored in compressed form 8 as illustrated in FIG. 4. During booting, the compressed Pkzip instructions 8 are read from the flash ROM 4 by the microprocessor 1, decompressed, and stored in the RAM 3 as decompressed instructions 8′ together with the decompressed operating instructions 6′ (FIG. 4). Alternatively, the compressed Pkzip instructions 8 may only be read from the flash ROM 4, and subsequently decompressed, when a specific request is made to alter the compressed instructions 6,8. In either case, the microprocessor 1 employs the decompressed Pkzip method to compress an amended version of the operating instructions 6′,8′ and writes the compressed instructions to the corresponding areas of the flash ROM.

Makinen, 4:18-32

See also Makinen, Figs. 2-4

1.3 decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form; and	Makinen, as evidenced by the example citations below, discloses "decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form; and"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the operating system relative to loading the operating system utilizing boot data in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Makinen discloses this limitation:

> A computer having a reduced instruction computer (RISC) architecture has a RISC central processing unit (CPU)(1) coupled to a RAM memory (3) and to a flash ROM memory (4). A set of compressed operating instructions (6,8), including a subset defining a compression method (8), are stored in the flash ROM (4) together with a set of uncompressed instructions (7) defining a compression algorithm. Upon booting of the computer, the uncompressed instructions (7) are read from the ROM (4) by the CPU (1) which then also reads the compressed instructions (6,8), decompresses them according to the decompression process (7), and writes the decompressed instructions (6′,8′) to the RAM (3). The compressed instructions (6,8) can be dynamically altered by the CPU (1), by generating an altered set of uncompressed instructions, compressing these in accordance with the now decompressed compression method (8′), and writing these to the flash ROM (4).

Makinen, Abstract

> At the heart of most modern electronic devices is a microprocessor or central processing unit which operates in accordance with a set of software operating instructions which together form an executable program. The instructions are stored in a digital memory which may be internal to the microprocessor or, as is more usually the case, externally connected to the microprocessor. The set of operating instructions generally define the basic input/output system (BIOS) of the microprocessor together with device drivers, libraries, and user applications.

Makinen, 1:10-19

> According to a first aspect of the present invention there is provided a method of operating apparatus having a central processing unit (CPU) with a reduced instruction set computer (RISC) architecture, and a read only memory (ROM), the method comprising reading a set of compressed RISC operating instructions from the ROM into the CPU, decompressing the compressed instructions in the CPU, and thereafter operating the apparatus in accordance with the decompressed instructions.

Makinen, 1:63-2:4

> According to a second aspect of the present invention there is provided a method of operating apparatus having a central processing unit (CPU) and a read only memory (ROM), the method comprising reading a set of compressed operating instructions from the ROM into the CPU, decompressing the compressed instructions in the CPU, and thereafter operating the apparatus in accordance with the decompressed instructions, the method further comprising generating one or more replacement or additional compressed instructions in the CPU and writing the compressed instruction(s) to the ROM.
>
> The above second aspect of the present invention makes it possible to amend the stored compressed instructions in a dynamic manner. This may, for example, allow a user to configure the computer according to his specific needs.
>
> Preferably, the method comprises the step of reading a set of operating instructions from the ROM into the CPU, which instructions define a program for compressing said replacement or additional instruction(s). More preferably, the instructions defining the compression program form part of said set of compressed operating instructions.

Makinen, 2:18-39

> Preferably, the method of the above first or second aspect of the invention comprises writing the decompressed instruction set to a random access memory (RAM). Thereafter, the decompressed instructions are read from the RAM by the CPU. It is noted that RAM typically offers high access speeds compared to slow (e.g. flash) ROM memory, giving a significant increase in system performance. In this case, increased speed also offers reduced power consumption compared

> to systems which use slow ROM memory and in which power is consumed even when the system is waiting to access the ROM.

Makinen, 2:40-50

> According to a third aspect of the present invention there is provided apparatus having a central processing unit (CPU) a reduced instruction set computer (RISC) architecture, and a read only memory (ROM), there being stored in the ROM a set of compressed RISC operating instructions, the CPU being arranged in use to read the compressed instructions from the ROM, to decompress these instructions, and subsequently to operate the apparatus in accordance with the decompressed instructions.

Makinen, 2:51-59

> According to a fourth aspect of the present invention there is provided apparatus comprising a central processing unit (CPU), a read only memory (ROM), and a set of compressed operating instructions stored in the ROM, the CPU being arranged in use to read the compressed instructions from the ROM, decompress the compressed instructions, and thereafter operate the apparatus in accordance with the decompressed instructions, the apparatus being further arranged in use to compress replacement or additional operating instructions and to write these compressed instructions to the ROM.

Makinen, 2:60-3:3:2

> The flash ROM 4 is used to store a set of RISC operating instructions which define the basic input/output system (BIOS) of the microprocessor as well as the device drivers, libraries, and user applications. The instructions are in compressed form, having previously been compressed using the Pkzip compression program

Makinen, 3:45-50

> In order to enable the microprocessor 1 to decompress the stored compressed instructions, the flash ROM 4 additionally stores a set of instructions, in uncompressed form, which define the Pkzip decompression program. FIG. 2 shows a memory map of the ROM 4 prior to booting the computer, where a part of the memory space is occupied by the compressed instructions 6 and a part is occupied by the uncompressed Pkzip instructions 7. In FIG. 2, the ROM 4 is shown coupled to the microprocessor 1, as is the RAM 3 which at this stage remains empty.

Makinen, 3:56-65

> Upon booting of the computer, the microprocessor 1 is directed by hardwired logic to read the first instruction of the decompressed instruction set 7, from the ROM 4. Thereafter the microprocessor is directed, by the decompressed set, to read and decompress the compressed instruction set 6. The expanded instruction set 6′ is then stored in the RAM 3 as shown in FIG. 3. The last operation that the set of decompression instructions perform is to cause the microprocessor 1 to jump to the first instruction in the decompressed code. Thereafter, the computer is operated in accordance with the decompressed operating instructions 6′.

Makinen, 3:66-4:9

> In order to reduce memory requirements, the corresponding Pkzip instructions may be stored in compressed form 8 as illustrated in FIG. 4. During booting, the compressed Pkzip instructions 8 are read from the flash ROM 4 by the microprocessor 1, decompressed, and stored in the RAM 3 as decompressed instructions 8′ together with the decompressed operating instructions 6′ (FIG. 4). Alternatively, the compressed Pkzip instructions 8 may only be read from the flash ROM 4, and subsequently decompressed, when a specific request is made to alter the compressed instructions 6,8. In either case, the microprocessor 1 employs the decompressed Pkzip method to compress an amended version of the operating instructions 6′,8′ and writes the compressed instructions to the corresponding areas of the flash ROM.

Makinen, 4:18-32

See also Makinen, Figs. 2-4

1.4.1 updating the boot data list,	Makinen, as evidenced by the example citations below, discloses "updating the boot data list,"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Makinen discloses this limitation:

> A computer having a reduced instruction computer (RISC) architecture has a RISC central processing unit (CPU)(1) coupled to a RAM memory (3) and to a flash ROM memory (4). A set of compressed operating instructions (6,8), including a subset defining a compression method (8), are stored in the flash ROM (4) together with a set of uncompressed instructions (7) defining a compression algorithm. Upon booting of the computer, the uncompressed instructions (7) are read from the ROM (4) by the CPU (1) which then also reads the compressed instructions (6,8), decompresses them according to the decompression process (7), and writes the decompressed instructions (6′,8′) to the RAM (3). The compressed instructions (6,8) can be dynamically altered by the CPU (1), by generating an altered set of uncompressed instructions, compressing these in accordance with the now decompressed compression method (8′), and writing these to the flash ROM (4).

Makinen, Abstract

> At the heart of most modern electronic devices is a microprocessor or central processing unit which operates in accordance with a set of software operating instructions which together form an executable program. The instructions are stored in a digital memory which may be internal to the microprocessor or, as is more usually the case, externally connected to the microprocessor. The set of operating instructions generally define the basic input/output system (BIOS) of the microprocessor together with device drivers, libraries, and user applications.

Makinen, 1:10-19

> According to a first aspect of the present invention there is provided a method of operating apparatus having a central processing unit (CPU) with a reduced instruction set computer (RISC) architecture, and a read

> only memory (ROM), the method comprising reading a set of compressed RISC operating instructions from the ROM into the CPU, decompressing the compressed instructions in the CPU, and thereafter operating the apparatus in accordance with the decompressed instructions.

Makinen, 1:63-2:4

> According to a second aspect of the present invention there is provided a method of operating apparatus having a central processing unit (CPU) and a read only memory (ROM), the method comprising reading a set of compressed operating instructions from the ROM into the CPU, decompressing the compressed instructions in the CPU, and thereafter operating the apparatus in accordance with the decompressed instructions, the method further comprising generating one or more replacement or additional compressed instructions in the CPU and writing the compressed instruction(s) to the ROM.
>
> The above second aspect of the present invention makes it possible to amend the stored compressed instructions in a dynamic manner. This may, for example, allow a user to configure the computer according to his specific needs.
>
> Preferably, the method comprises the step of reading a set of operating instructions from the ROM into the CPU, which instructions define a program for compressing said replacement or additional instruction(s). More preferably, the instructions defining the compression program form part of said set of compressed operating instructions.

Makinen, 2:18-39

> Preferably, the method of the above first or second aspect of the invention comprises writing the decompressed instruction set to a random access memory (RAM). Thereafter, the decompressed instructions are read from the RAM by the CPU. It is noted that RAM typically offers high access speeds compared to slow (e.g. flash) ROM memory, giving a significant increase in system performance. In this case, increased speed also offers reduced power consumption compared to systems which use slow ROM memory and in which power is consumed even when the system is waiting to access the ROM.

Makinen, 2:40-50

> According to a third aspect of the present invention there is provided apparatus having a central processing unit (CPU) a reduced instruction set computer (RISC) architecture, and a read only memory (ROM),

> there being stored in the ROM a set of compressed RISC operating instructions, the CPU being arranged in use to read the compressed instructions from the ROM, to decompress these instructions, and subsequently to operate the apparatus in accordance with the decompressed instructions.

Makinen, 2:51-59

> According to a fourth aspect of the present invention there is provided apparatus comprising a central processing unit (CPU), a read only memory (ROM), and a set of compressed operating instructions stored in the ROM, the CPU being arranged in use to read the compressed instructions from the ROM, decompress the compressed instructions, and thereafter operate the apparatus in accordance with the decompressed instructions, the apparatus being further arranged in use to compress replacement or additional operating instructions and to write these compressed instructions to the ROM.

Makinen, 2:60-3:3:2

> It will be appreciated that the operating system of the computer may be altered by directly accessing the flash ROM 5 to erase and/or rewrite compressed instructions 6 stored therein. However, in some circumstances it may be desirable for the end-user to be able to alter the compressed operating instructions 6, or indeed for the computer itself to be able to 'dynamically' alter the instructions. To this end, the Pkzip compression method may also be stored in the flash ROM 4.

Makinen, 4:10-17

> In order to reduce memory requirements, the corresponding Pkzip instructions may be stored in compressed form 8 as illustrated in FIG. 4. During booting, the compressed Pkzip instructions 8 are read from the flash ROM 4 by the microprocessor 1, decompressed, and stored in the RAM 3 as decompressed instructions 8′ together with the decompressed operating instructions 6′ (FIG. 4). Alternatively, the compressed Pkzip instructions 8 may only be read from the flash ROM 4, and subsequently decompressed, when a specific request is made to alter the compressed instructions 6,8. In either case, the microprocessor 1 employs the decompressed Pkzip method to compress an amended version of the operating instructions 6′, 8′ and writes the compressed instructions to the corresponding areas of the flash ROM.

Makinen, 4:18-32

See also Makinen, Figs. 2-4

1.4.2 wherein the decompressed portion of boot data comprises a portion of the operating system.	Makinen, as evidenced by the example citations below, discloses "wherein the decompressed portion of boot data comprises a portion of the operating system."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the decompressed portion of boot data comprises a portion of the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Makinen discloses this limitation:

> A computer having a reduced instruction computer (RISC) architecture has a RISC central processing unit (CPU)(1) coupled to a RAM memory (3) and to a flash ROM memory (4). A set of compressed operating instructions (6,8), including a subset defining a compression method (8), are stored in the flash ROM (4) together with a set of uncompressed instructions (7) defining a compression algorithm. Upon booting of the computer, the uncompressed instructions (7) are read from the ROM (4) by the CPU (1) which then also reads the compressed instructions (6,8), decompresses them according to the decompression process (7), and writes the decompressed instructions (6′,8′) to the RAM (3). The compressed instructions (6,8) can be dynamically altered by the CPU (1), by generating an altered set of uncompressed instructions, compressing these in accordance with the now decompressed compression method (8′), and writing these to the flash ROM (4).

Makinen, Abstract

> At the heart of most modern electronic devices is a microprocessor or central processing unit which operates in accordance with a set of software operating instructions which together form an executable program. The instructions are stored in a digital memory which may be internal to the microprocessor or, as is more usually the case, externally connected to the microprocessor. The set of operating instructions generally define the basic input/output system (BIOS) of the microprocessor together with device drivers, libraries, and user applications.

Makinen, 1:10-19

> According to a second aspect of the present invention there is provided a method of operating apparatus having a central processing unit (CPU) and a read only memory (ROM), the method comprising reading a set of compressed operating instructions from the ROM into the CPU, decompressing the compressed instructions in the CPU, and thereafter operating the apparatus in accordance with the decompressed instructions, the method further comprising generating one or more replacement or additional compressed instructions in the CPU and writing the compressed instruction(s) to the ROM.
>
> The above second aspect of the present invention makes it possible to amend the stored compressed instructions in a dynamic manner. This may, for example, allow a user to configure the computer according to his specific needs.
>
> Preferably, the method comprises the step of reading a set of operating instructions from the ROM into the CPU, which instructions define a program for compressing said replacement or additional instruction(s). More preferably, the instructions defining the compression program form part of said set of compressed operating instructions.

Makinen, 2:18-39

> Preferably, the method of the above first or second aspect of the invention comprises writing the decompressed instruction set to a random access memory (RAM). Thereafter, the decompressed instructions are read from the RAM by the CPU. It is noted that RAM typically offers high access speeds compared to slow (e.g. flash) ROM memory, giving a significant increase in system performance. In this case, increased speed also offers reduced power consumption compared to systems which use slow ROM memory and in which power is consumed even when the system is waiting to access the ROM.

Makinen, 2:40-50

> According to a third aspect of the present invention there is provided apparatus having a central processing unit (CPU) a reduced instruction set computer (RISC) architecture, and a read only memory (ROM), there being stored in the ROM a set of compressed RISC operating instructions, the CPU being arranged in use to read the compressed instructions from the ROM, to decompress these instructions, and subsequently to operate the apparatus in accordance with the decompressed instructions.

Makinen, 2:51-59

According to a fourth aspect of the present invention there is provided apparatus comprising a central processing unit (CPU), a read only memory (ROM), and a set of compressed operating instructions stored in the ROM, the CPU being arranged in use to read the compressed instructions from the ROM, decompress the compressed instructions, and thereafter operate the apparatus in accordance with the decompressed instructions, the apparatus being further arranged in use to compress replacement or additional operating instructions and to write these compressed instructions to the ROM. Makinen, 2:60-3:3:2

2. The method of claim 1, wherein the updating comprises: associating additional boot data with the boot data list.	Makinen, as evidenced by the example citations below, discloses "wherein the updating comprises: associating additional boot data with the boot data list."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein associating additional boot data with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Makinen discloses this limitation:

See Claim 1.4.1 above.

Upon booting of the computer, the microprocessor 1 is directed by hardwired logic to read the first instruction of the decompressed instruction set 7, from the ROM 4. Thereafter the microprocessor is directed, by the decompressed set, to read and decompress the compressed instruction set 6. The expanded instruction set 6′ is then stored in the RAM 3 as shown in FIG. 3. The last operation that the set of decompression instructions perform is to cause the microprocessor 1 to jump to the first instruction in the decompressed code. Thereafter, the computer is operated in accordance with the decompressed operating instructions 6′.

Makinen 3:66-4:28

In order to reduce memory requirements, the corresponding Pkzip instructions may be stored in compressed form 8 as illustrated in FIG. 4. During booting, the compressed Pkzip instructions 8 are read from the flash ROM 4 by the microprocessor 1, decompressed, and stored in the RAM 3 as decompressed instructions 8′ together with the decompressed operating instructions 6′ (FIG. 4). Alternatively, the compressed Pkzip instructions 8 may only be read from the flash ROM 4, and subsequently decompressed, when a specific request is made to alter the compressed instructions 6,8. In either case, the microprocessor 1 employs the decompressed Pkzip method to compress an amended version of the operating instructions 6′, 8′ and writes the compressed instructions to the corresponding areas of the flash ROM.

Makinen 4:18-31

The above second aspect of the present invention makes it possible to amend the stored compressed instructions in a dynamic manner. This may, for example, allow a user to configure the computer according to his specific needs.

Preferably, the method comprises the step of reading a set of operating instructions from the ROM into the CPU, which instructions define a program for compressing said replacement or additional instruction(s). More preferably, the instructions defining the compression program form part of said set of compressed operating instructions. Makinen, 2:29-39

3. The method of claim 1, wherein the updating comprises: removing an association of additional boot data that is associated with the boot data list from the boot data list.	Makinen, as evidenced by the example citations below, discloses "wherein the updating comprises: removing an association of additional boot data that is associated with the boot data list from the boot data list."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein removing an association of additional boot data that is associated with the boot data list from the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Makinen discloses this limitation:

See Claim 1.4.1 above.

Upon booting of the computer, the microprocessor 1 is directed by hardwired logic to read the first instruction of the decompressed instruction set 7, from the ROM 4. Thereafter the microprocessor is directed, by the decompressed set, to read and decompress the compressed instruction set 6. The expanded instruction set 6′ is then stored in the RAM 3 as shown in FIG. 3. The last operation that the set of decompression instructions perform is to cause the microprocessor 1 to jump to the first instruction in the decompressed code. Thereafter, the computer is operated in accordance with the decompressed operating instructions 6′.

Makinen 3:66-4:28

In order to reduce memory requirements, the corresponding Pkzip instructions may be stored in compressed form 8 as illustrated in FIG. 4. During booting, the compressed Pkzip instructions 8 are read from the flash ROM 4 by the microprocessor 1, decompressed, and stored in the RAM 3 as decompressed instructions 8′ together with the decompressed operating instructions 6′ (FIG. 4). Alternatively, the compressed Pkzip instructions 8 may only be read from the flash ROM 4, and subsequently decompressed, when a specific request is made to alter the compressed instructions 6,8. In either case, the microprocessor 1 employs the decompressed Pkzip method to compress an amended version of the operating instructions 6′,8′ and writes the compressed instructions to the corresponding areas of the flash ROM.

Makinen 4:18-31

The above second aspect of the present invention makes it possible to amend the stored compressed instructions in a dynamic manner. This may, for example, allow a user to configure the computer according to his specific needs.

Preferably, the method comprises the step of reading a set of operating instructions from the ROM into the CPU, which instructions define a program for compressing said replacement or additional instruction(s). More preferably, the instructions defining the compression program form part of said set of compressed operating instructions.

Makinen, 2:29-39

4. The method of claim 1, wherein the updating comprises: associating additional boot data with the boot data list; and compressing a portion of the additional boot data.	Makinen, as evidenced by the example citations below, discloses "wherein the updating comprises: associating additional boot data with the boot data list; and compressing a portion of the additional boot data."

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating wherein associating additional boot data with the boot data list; and compressing a portion of the additional boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Makinen discloses this limitation:

See Claims 1.4.1, and 2 above.

Upon booting of the computer, the microprocessor 1 is directed by hardwired logic to read the first instruction of the decompressed instruction set 7, from the ROM 4. Thereafter the microprocessor is directed, by the decompressed set, to read and decompress the compressed instruction set 6. The expanded instruction set 6′ is then stored in the RAM 3 as shown in FIG. 3. The last operation that the set of decompression instructions perform is to cause the microprocessor 1 to jump to the first instruction in the decompressed code. Thereafter, the computer is operated in accordance with the decompressed operating instructions 6′.

Makinen 3:66-4:28

In order to reduce memory requirements, the corresponding Pkzip instructions may be stored in compressed form 8 as illustrated in FIG. 4. During booting, the compressed Pkzip instructions 8 are read from the flash ROM 4 by the microprocessor 1, decompressed, and stored in the RAM 3 as decompressed instructions 8′ together with the decompressed operating instructions 6′ (FIG. 4). Alternatively, the compressed Pkzip instructions 8 may only be read from the flash ROM 4, and subsequently decompressed, when a specific request is made to alter the compressed instructions 6,8. In either case, the microprocessor 1 employs the decompressed Pkzip method to compress an amended version of the operating instructions 6′, 8′ and writes the compressed instructions to the corresponding areas of the flash ROM.

Makinen 4:18-31

5 (Preamble) A method for booting a computer system, the method comprising:	Makinen, as evidenced by the example citations below, discloses "a method for booting a computer system, the method comprising:"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, booting a computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Makinen discloses this limitation:

See Claims 1-1.4.2 above.

> A computer having a reduced instruction computer (RISC) architecture has a RISC central processing unit (CPU)(1) coupled to a RAM memory (3) and to a flash ROM memory (4). A set of compressed operating instructions (6,8), including a subset defining a compression method (8), are stored in the flash ROM (4) together with a set of uncompressed instructions (7) defining a compression algorithm. Upon booting of the computer, the uncompressed instructions (7) are read from the ROM (4) by the CPU (1) which then also reads the compressed instructions (6,8), decompresses them according to the decompression process (7), and writes the decompressed instructions (6′,8′) to the RAM (3). The compressed instructions (6,8) can be dynamically altered by the CPU (1), by generating an altered set of uncompressed instructions, compressing these in accordance with the now decompressed compression method (8′), and writing these to the flash ROM (4).

Makinen, Abstract

> According to a first aspect of the present invention there is provided a method of operating apparatus having a central processing unit (CPU) with a reduced instruction set computer (RISC) architecture, and a read only memory (ROM), the method comprising reading a set of compressed RISC operating instructions from the ROM into the CPU, decompressing the compressed instructions in the CPU, and thereafter operating the apparatus in accordance with the decompressed instructions.

Makinen, 1:63-2:4

> According to a second aspect of the present invention there is provided a method of operating apparatus having a central processing unit (CPU) and a read only memory (ROM), the method comprising reading a set of

> compressed operating instructions from the ROM into the CPU, decompressing the compressed instructions in the CPU, and thereafter operating the apparatus in accordance with the decompressed instructions, the method further comprising generating one or more replacement or additional compressed instructions in the CPU and writing the compressed instruction(s) to the ROM.
>
> The above second aspect of the present invention makes it possible to amend the stored compressed instructions in a dynamic manner. This may, for example, allow a user to configure the computer according to his specific needs.
>
> Preferably, the method comprises the step of reading a set of operating instructions from the ROM into the CPU, which instructions define a program for compressing said replacement or additional instruction(s). More preferably, the instructions defining the compression program form part of said set of compressed operating instructions.

Makinen, 2:18-39

> Preferably, the method of the above first or second aspect of the invention comprises writing the decompressed instruction set to a random access memory (RAM). Thereafter, the decompressed instructions are read from the RAM by the CPU. It is noted that RAM typically offers high access speeds compared to slow (e.g. flash) ROM memory, giving a significant increase in system performance. In this case, increased speed also offers reduced power consumption compared to systems which use slow ROM memory and in which power is consumed even when the system is waiting to access the ROM.

Makinen, 2:40-50

> According to a third aspect of the present invention there is provided apparatus having a central processing unit (CPU) a reduced instruction set computer (RISC) architecture, and a read only memory (ROM), there being stored in the ROM a set of compressed RISC operating instructions, the CPU being arranged in use to read the compressed instructions from the ROM, to decompress these instructions, and subsequently to operate the apparatus in accordance with the decompressed instructions.

Makinen, 2:51-59

> According to a fourth aspect of the present invention there is provided apparatus comprising a central processing unit (CPU), a read only memory (ROM), and a set of compressed operating instructions stored

> in the ROM, the CPU being arranged in use to read the compressed instructions from the ROM, decompress the compressed instructions, and thereafter operate the apparatus in accordance with the decompressed instructions, the apparatus being further arranged in use to compress replacement or additional operating instructions and to write these compressed instructions to the ROM.

Makinen, 2:60-3:3:2

> The flash ROM 4 is used to store a set of RISC operating instructions which define the basic input/output system (BIOS) of the microprocessor as well as the device drivers, libraries, and user applications. The instructions are in compressed form, having previously been compressed using the Pkzip compression program

Makinen, 3:45-50

> It will be appreciated that the operating system of the computer may be altered by directly accessing the flash ROM 5 to erase and/or rewrite compressed instructions 6 stored therein. However, in some circumstances it may be desirable for the end-user to be able to alter the compressed operating instructions 6, or indeed for the computer itself to be able to 'dynamically' alter the instructions. To this end, the Pkzip compression method may also be stored in the flash ROM 4.

Makinen, 4:10-17

> A computer having a reduced instruction computer (RISC) architecture has a RISC central processing unit (CPU)(1) coupled to a RAM memory (3) and to a flash ROM memory (4). A set of compressed operating instructions (6,8), including a subset defining a compression method (8), are stored in the flash ROM (4) together with a set of uncompressed instructions (7) defining a compression algorithm. Upon booting of the computer, the uncompressed instructions (7) are read from the ROM (4) by the CPU (1) which then also reads the compressed instructions (6,8), decompresses them according to the decompression process (7), and writes the decompressed instructions (6′,8′) to the RAM (3). The compressed instructions (6,8) can be dynamically altered by the CPU (1), by generating an altered set of uncompressed instructions, compressing these in accordance with the now decompressed compression method (8′), and writing these to the flash ROM (4).

Makinen, 1:10-19

> Upon booting of the computer, the microprocessor 1 is directed by hardwired logic to read the first instruction of the decompressed

> instruction set 7, from the ROM 4. Thereafter the microprocessor is directed, by the decompressed set, to read and decompress the compressed instruction set 6. The expanded instruction set 6′ is then stored in the RAM 3 as shown in FIG. 3. The last operation that the set of decompression instructions perform is to cause the microprocessor 1 to jump to the first instruction in the decompressed code. Thereafter, the computer is operated in accordance with the decompressed operating instructions 6′.

Makinen, 3:66-4:9

See also Makinen, Fig. 3

Appendix C13
Invalidity of U.S. Patent 8,880,862 based on Makinen

5.1 storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory;	Makinen, as evidenced by the example citations below, discloses “storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory”

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing boot data in a compressed form that is associated with a portion of a boot data list in a first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions.

Makinen discloses this limitation:

See Claim 1.1 above.

> The present invention relates to executable programs and in particular to a method and apparatus for storing and using executable programs.

Makinen, 1:5-7

> A compressed program is stored in the external memory unit and, before commencing processing operations, the compressed program is read from the external memory unit in blocks and decompressed. The decompressed program is then written to the main memory unit. However, the compression ratio which can be achieved with CISC code is relatively low and it is not believed that the disclosure of JP-131848 has been widely used.

Makinen, 1:47-54

> Preferably, the apparatus comprises a random access memory which, in use, is arranged to store the operating instructions following decompression by the CPU.

Makinen, 3:4-11

> In order to reduce memory requirements, the corresponding Pkzip instructions may be stored in compressed form 8 as illustrated in FIG. 4. During booting, the compressed Pkzip instructions 8 are read from the flash ROM 4 by the microprocessor 1, decompressed, and stored in the RAM 3 as decompressed instructions 8′ together with the decompressed operating instructions 6′ (FIG. 4). Alternatively, the compressed Pkzip instructions 8 may only be read from the flash ROM 4, and subsequently decompressed, when a specific request is made to alter the compressed instructions 6,8. In either case, the microprocessor 1 employs the decompressed Pkzip method to compress an amended version of

> the operating instructions 6′,8′ and writes the compressed instructions to the corresponding areas of the flash ROM.

Makinen, 4:18-32

5.2 loading the stored compressed boot data from the first memory;	Makinen, as evidenced by the example citations below, discloses "loading the stored compressed boot data from the first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the stored compressed boot data from the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claim 1.1 above.	

5.3 accessing the loaded compressed boot data;	Makinen, as evidenced by the example citations below, discloses “accessing the loaded compressed boot data”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Makinen discloses this limitation: *See* Claim 1.2 above.	

5.4 decompressing the accessed compressed boot data;	Makinen, as evidenced by the example citations below, discloses “decompressing the accessed compressed boot data”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed compressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Makinen discloses this limitation: *See* Claim 1.3 above.	

5.5 utilizing the decompressed boot data to at least partially boot the computer system;	Makinen, as evidenced by the example citations below, discloses "utilizing the decompressed boot data to at least partially boot the computer system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed boot data to at least partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claims 1.3 and 1.4.2 above. > A computer having a reduced instruction computer (RISC) architecture has a RISC central processing unit (CPU)(1) coupled to a RAM memory (3) and to a flash ROM memory (4). A set of compressed operating instructions (6,8), including a subset defining a compression method (8), are stored in the flash ROM (4) together with a set of uncompressed instructions (7) defining a compression algorithm. Upon booting of the computer, the uncompressed instructions (7) are read from the ROM (4) by the CPU (1) which then also reads the compressed instructions (6,8), decompresses them according to the decompression process (7), and writes the decompressed instructions (6′,8′) to the RAM (3). The compressed instructions (6,8) can be dynamically altered by the CPU (1), by generating an altered set of uncompressed instructions, compressing these in accordance with the now decompressed compression method (8′), and writing these to the flash ROM (4). Makinen, Abstract > At the heart of most modern electronic devices is a microprocessor or central processing unit which operates in accordance with a set of software operating instructions which together form an executable program. The instructions are stored in a digital memory which may be internal to the microprocessor or, as is more usually the case, externally connected to the microprocessor. The set of operating instructions generally define the basic input/output system (BIOS) of the microprocessor together with device drivers, libraries, and user applications. Makinen, 1:10-19	

> According to a first aspect of the present invention there is provided a method of operating apparatus having a central processing unit (CPU) with a reduced instruction set computer (RISC) architecture, and a read only memory (ROM), the method comprising reading a set of compressed RISC operating instructions from the ROM into the CPU, decompressing the compressed instructions in the CPU, and thereafter operating the apparatus in accordance with the decompressed instructions.

Makinen, 1:63-2:4

> According to a second aspect of the present invention there is provided a method of operating apparatus having a central processing unit (CPU) and a read only memory (ROM), the method comprising reading a set of compressed operating instructions from the ROM into the CPU, decompressing the compressed instructions in the CPU, and thereafter operating the apparatus in accordance with the decompressed instructions, the method further comprising generating one or more replacement or additional compressed instructions in the CPU and writing the compressed instruction(s) to the ROM.
>
> The above second aspect of the present invention makes it possible to amend the stored compressed instructions in a dynamic manner. This may, for example, allow a user to configure the computer according to his specific needs.
>
> Preferably, the method comprises the step of reading a set of operating instructions from the ROM into the CPU, which instructions define a program for compressing said replacement or additional instruction(s). More preferably, the instructions defining the compression program form part of said set of compressed operating instructions.

Makinen, 2:18-39

> Preferably, the method of the above first or second aspect of the invention comprises writing the decompressed instruction set to a random access memory (RAM). Thereafter, the decompressed instructions are read from the RAM by the CPU. It is noted that RAM typically offers high access speeds compared to slow (e.g. flash) ROM memory, giving a significant increase in system performance. In this case, increased speed also offers reduced power consumption compared to systems which use slow ROM memory and in which power is consumed even when the system is waiting to access the ROM.

Makinen, 2:40-50

> According to a third aspect of the present invention there is provided apparatus having a central processing unit (CPU) a reduced instruction set computer (RISC) architecture, and a read only memory (ROM), there being stored in the ROM a set of compressed RISC operating instructions, the CPU being arranged in use to read the compressed instructions from the ROM, to decompress these instructions, and subsequently to operate the apparatus in accordance with the decompressed instructions.

Makinen, 2:51-59

> According to a fourth aspect of the present invention there is provided apparatus comprising a central processing unit (CPU), a read only memory (ROM), and a set of compressed operating instructions stored in the ROM, the CPU being arranged in use to read the compressed instructions from the ROM, decompress the compressed instructions, and thereafter operate the apparatus in accordance with the decompressed instructions, the apparatus being further arranged in use to compress replacement or additional operating instructions and to write these compressed instructions to the ROM.

Makinen, 2:60-3:3:2

> The flash ROM 4 is used to store a set of RISC operating instructions which define the basic input/output system (BIOS) of the microprocessor as well as the device drivers, libraries, and user applications. The instructions are in compressed form, having previously been compressed using the Pkzip compression program

Makinen, 3:45-50

> In order to enable the microprocessor 1 to decompress the stored compressed instructions, the flash ROM 4 additionally stores a set of instructions, in uncompressed form, which define the Pkzip decompression program. FIG. 2 shows a memory map of the ROM 4 prior to booting the computer, where a part of the memory space is occupied by the compressed instructions 6 and a part is occupied by the uncompressed Pkzip instructions 7. In FIG. 2, the ROM 4 is shown coupled to the microprocessor 1, as is the RAM 3 which at this stage remains empty.

Makinen, 3:56-65

> Upon booting of the computer, the microprocessor 1 is directed by hardwired logic to read the first instruction of the decompressed instruction set 7, from the ROM 4. Thereafter the microprocessor is

> directed, by the decompressed set, to read and decompress the compressed instruction set 6. The expanded instruction set 6′ is then stored in the RAM 3 as shown in FIG. 3. The last operation that the set of decompression instructions perform is to cause the microprocessor 1 to jump to the first instruction in the decompressed code. Thereafter, the computer is operated in accordance with the decompressed operating instructions 6′.

Makinen, 3:66-4:9

> In order to reduce memory requirements, the corresponding Pkzip instructions may be stored in compressed form 8 as illustrated in FIG. 4. During booting, the compressed Pkzip instructions 8 are read from the flash ROM 4 by the microprocessor 1, decompressed, and stored in the RAM 3 as decompressed instructions 8′ together with the decompressed operating instructions 6′ (FIG. 4). Alternatively, the compressed Pkzip instructions 8 may only be read from the flash ROM 4, and subsequently decompressed, when a specific request is made to alter the compressed instructions 6,8. In either case, the microprocessor 1 employs the decompressed Pkzip method to compress an amended version of the operating instructions 6′,8′ and writes the compressed instructions to the corresponding areas of the flash ROM.

Makinen, 4:18-32

See also Makinen, Figs. 2-4

5.6 updating the boot data list;	Makinen, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claim 1.4.1 above.	

5.7 wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form.	Makinen, as evidenced by the example citations below, discloses "wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the loading, the accessing, and the decompressing occur within a period of time which is less than a time to access the boot data from the first memory if the boot data was stored in the first memory in an uncompressed form.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Makinen discloses this limitation:

See Claims 1-1.3 above.

> Compared to conventional ROM, flash ROM remains expensive. There therefore exists a desire to reduce the amount of flash ROM used in individual products. Furthermore, accessing flash ROM is relatively slow, significantly reducing the performance of electronic devices. Conventional computer architectures rely upon a complex instruction set computer (CISC) architecture. This utilises a large number (e.g. 1000) of instructions which define very specific tasks. The instructions are of variable length and are decoded by the computer's CPU before execution. There is described in Japanese non-examined patent publication no. 55-131848 a data processing unit which comprises a central processing unit, a main memory unit, and an external memory unit. A compressed program is stored in the external memory unit and, before commencing processing operations, the compressed program is read from the external memory unit in blocks and decompressed. The decompressed program is then written to the main memory unit. However, the compression ratio which can be achieved with CISC code is relatively low and it is not believed that the disclosure of JP-131848 has been widely used.

Makinen, 1:34-54

> Preferably, the method of the above first or second aspect of the invention comprises writing the decompressed instruction set to a random access memory (RAM). Thereafter, the decompressed instructions are read from the RAM by the CPU. It is noted that RAM typically offers high access speeds compared to

slow (e.g. flash) ROM memory, giving a significant increase in system performance. In this case, increased speed also offers reduced power consumption compared to systems which use slow ROM memory and in which power is consumed even when the system is waiting to access the ROM. Makinen, 2:40-50

6 (Preamble) A system comprising: a processor;	Makinen, as evidenced by the example citations below, discloses "a system comprising: a processor:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a system comprising: a processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* ***<u>Add disclosure from '608 Patent Claim 1.2</u>***	

6.1 a memory; and	Makinen, as evidenced by the example citations below, discloses "a memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a memory.), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claims 1.1 and 1.2 above.	

6.2 a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor	Makinen, as evidenced by the example citations below, discloses "a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a second memory configured to store boot data in a compressed form for booting the system and a logic code associated with the processor), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claims 1.1, 1.2, and Claim 6 (Preamble) above. *See also* ***<u>Look for additional references where there is a second memory configured to store compressed boot data before it is loaded into the first memory (note: Claim 6.4 below loads from second memory into first memory)</u>*** Compared to conventional ROM, flash ROM remains expensive. There therefore exists a desire to reduce the amount of flash ROM used in individual products. Furthermore, accessing flash ROM is relatively slow, significantly reducing the performance of electronic devices. Conventional computer architectures rely upon a complex instruction set computer (CISC) architecture. This utilises a large number (e.g. 1000) of instructions which define very specific tasks. The instructions are of variable length and are decoded by the computer's CPU before execution. There is described in Japanese non-examined patent publication no. 55-131848 a data processing unit which comprises a central processing unit, a main memory unit, and an external memory unit. A compressed program is stored in the external memory unit and, before commencing processing operations, the compressed program is read from the external memory unit in blocks and decompressed. The decompressed program is then written to the main memory unit. However, the compression ratio which can be achieved with CISC code is relatively low and it is not believed that the disclosure of JP-131848 has been widely used. Makinen 1:34-55	

6.3 wherein the processor is configured:	Makinen, as evidenced by the example citations below, discloses "wherein the processor is configured:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the processor is configured), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claim 6 (Preamble) above	

6.4 to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory,	Makinen, as evidenced by the example citations below, discloses "to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to load a portion of the boot data in the compressed form that is associated with a boot data list used for booting the system into the first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claim 1.1 above.	

6.5 to access the loaded portion of the boot data in the compressed form,	Makinen, as evidenced by the example citations below, discloses “to access the loaded portion of the boot data in the compressed form”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to access the loaded portion of the boot data in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Makinen discloses this limitation: *See* Claim 1.2 above.	

6.6 to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data, and	Makinen, as evidenced by the example citations below, discloses “to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to decompress the accessed portion of the boot data in the compressed form at a rate that decreases a boot time of the system relative to booting the system with uncompressed boot data), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Makinen discloses this limitation: *See* Claim 1.3 above.	

6.7 to update the boot data list.	Makinen, as evidenced by the example citations below, discloses "to update the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, to update the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claim 1.4.1 above.	

Appendix C13
Invalidity of U.S. Patent 8,880,862 based on Makinen

8 (Preamble) A method of loading an operating system for booting a computer system, comprising:	Makinen, as evidenced by the example citations below, discloses "A method of loading an operating system for booting a computer system, comprising:"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, a method of loading an operating system for booting a computer system, comprising), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claim 1 (Preamble) above.	

Appendix C13
Invalidity of U.S. Patent 8,880,862 based on Makinen

8.1 storing a portion of the operating system in a compressed form in a first memory;	Makinen, as evidenced by the example citations below, discloses “storing a portion of the operating system in a compressed form in a first memory”
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, storing a portion of the operating system in a compressed form in a first memory), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple’s Invalidity Contentions. Makinen discloses this limitation: *See* Claim 6.2 above.	

8.2 loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list;	Makinen, as evidenced by the example citations below, discloses "loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, loading the portion of the operating system from the first memory to a second memory, the portion of the operating system being associated with a boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claim 1.1 above.	

8.3 accessing the loaded portion of the operating system from the second memory in the compressed form;	Makinen, as evidenced by the example citations below, discloses "accessing the loaded portion of the operating system from the second memory in the compressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, accessing the loaded portion of the operating system from the second memory in the compressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claim 1.2 above.	

8.4 decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system;	Makinen, as evidenced by the example citations below, discloses "decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, decompressing the accessed portion of the operating system to provide a decompressed portion of the operating system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claims 1.3 and 1.4.2 above.	

8.5 utilizing the decompressed portion of the operating system to at least partially boot the computer system; and	Makinen, as evidenced by the example citations below, discloses "utilizing the decompressed portion of the operating system to at least partially boot the computer system"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, utilizing the decompressed portion of the operating system to at least partially boot the computer system), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Makinen discloses this limitation:

See Claims 1.3 and 1.4.2 above.

> A computer having a reduced instruction computer (RISC) architecture has a RISC central processing unit (CPU)(1) coupled to a RAM memory (3) and to a flash ROM memory (4). A set of compressed operating instructions (6,8), including a subset defining a compression method (8), are stored in the flash ROM (4) together with a set of uncompressed instructions (7) defining a compression algorithm. Upon booting of the computer, the uncompressed instructions (7) are read from the ROM (4) by the CPU (1) which then also reads the compressed instructions (6,8), decompresses them according to the decompression process (7), and writes the decompressed instructions (6′,8′) to the RAM (3). The compressed instructions (6,8) can be dynamically altered by the CPU (1), by generating an altered set of uncompressed instructions, compressing these in accordance with the now decompressed compression method (8′), and writing these to the flash ROM (4).

Makinen, Abstract

> At the heart of most modern electronic devices is a microprocessor or central processing unit which operates in accordance with a set of software operating instructions which together form an executable program. The instructions are stored in a digital memory which may be internal to the microprocessor or, as is more usually the case, externally connected to the microprocessor. The set of operating instructions generally define the basic input/output system (BIOS) of the microprocessor together with device drivers, libraries, and user applications.

Makinen, 1:10-19

> According to a first aspect of the present invention there is provided a method of operating apparatus having a central processing unit (CPU) with a reduced instruction set computer (RISC) architecture, and a read only memory (ROM), the method comprising reading a set of compressed RISC operating instructions from the ROM into the CPU, decompressing the compressed instructions in the CPU, and thereafter operating the apparatus in accordance with the decompressed instructions.

Makinen, 1:63-2:4

> According to a second aspect of the present invention there is provided a method of operating apparatus having a central processing unit (CPU) and a read only memory (ROM), the method comprising reading a set of compressed operating instructions from the ROM into the CPU, decompressing the compressed instructions in the CPU, and thereafter operating the apparatus in accordance with the decompressed instructions, the method further comprising generating one or more replacement or additional compressed instructions in the CPU and writing the compressed instruction(s) to the ROM.
>
> The above second aspect of the present invention makes it possible to amend the stored compressed instructions in a dynamic manner. This may, for example, allow a user to configure the computer according to his specific needs.
>
> Preferably, the method comprises the step of reading a set of operating instructions from the ROM into the CPU, which instructions define a program for compressing said replacement or additional instruction(s). More preferably, the instructions defining the compression program form part of said set of compressed operating instructions.

Makinen, 2:18-39

> Preferably, the method of the above first or second aspect of the invention comprises writing the decompressed instruction set to a random access memory (RAM). Thereafter, the decompressed instructions are read from the RAM by the CPU. It is noted that RAM typically offers high access speeds compared to slow (e.g. flash) ROM memory, giving a significant increase in system performance. In this case, increased speed also offers reduced power consumption compared to systems which use slow ROM memory and in which power is consumed even when the system is waiting to access the ROM.

Makinen, 2:40-50

> According to a third aspect of the present invention there is provided apparatus having a central processing unit (CPU) a reduced instruction set computer (RISC) architecture, and a read only memory (ROM), there being stored in the ROM a set of compressed RISC operating instructions, the CPU being arranged in use to read the compressed instructions from the ROM, to decompress these instructions, and subsequently to operate the apparatus in accordance with the decompressed instructions.

Makinen, 2:51-59

> According to a fourth aspect of the present invention there is provided apparatus comprising a central processing unit (CPU), a read only memory (ROM), and a set of compressed operating instructions stored in the ROM, the CPU being arranged in use to read the compressed instructions from the ROM, decompress the compressed instructions, and thereafter operate the apparatus in accordance with the decompressed instructions, the apparatus being further arranged in use to compress replacement or additional operating instructions and to write these compressed instructions to the ROM.

Makinen, 2:60-3:3:2

> The flash ROM 4 is used to store a set of RISC operating instructions which define the basic input/output system (BIOS) of the microprocessor as well as the device drivers, libraries, and user applications. The instructions are in compressed form, having previously been compressed using the Pkzip compression program

Makinen, 3:45-50

> In order to enable the microprocessor 1 to decompress the stored compressed instructions, the flash ROM 4 additionally stores a set of instructions, in uncompressed form, which define the Pkzip decompression program. FIG. 2 shows a memory map of the ROM 4 prior to booting the computer, where a part of the memory space is occupied by the compressed instructions 6 and a part is occupied by the uncompressed Pkzip instructions 7. In FIG. 2, the ROM 4 is shown coupled to the microprocessor 1, as is the RAM 3 which at this stage remains empty.

Makinen, 3:56-65

> Upon booting of the computer, the microprocessor 1 is directed by hardwired logic to read the first instruction of the decompressed instruction set 7, from the ROM 4. Thereafter the microprocessor is directed, by the decompressed set, to read and decompress the compressed instruction set 6. The expanded instruction set 6′ is then stored in the RAM 3 as shown in FIG. 3. The last operation that the set of decompression instructions perform is to cause the microprocessor 1 to jump to the first instruction in the decompressed code. Thereafter, the computer is operated in accordance with the decompressed operating instructions 6′.

Makinen, 3:66-4:9

> In order to reduce memory requirements, the corresponding Pkzip instructions may be stored in compressed form 8 as illustrated in FIG. 4. During booting, the compressed Pkzip instructions 8 are read from the flash ROM 4 by the microprocessor 1, decompressed, and stored in the RAM 3 as decompressed instructions 8′ together with the decompressed operating instructions 6′ (FIG. 4). Alternatively, the compressed Pkzip instructions 8 may only be read from the flash ROM 4, and subsequently decompressed, when a specific request is made to alter the compressed instructions 6, 8. In either case, the microprocessor 1 employs the decompressed Pkzip method to compress an amended version of the operating instructions 6′, 8′ and writes the compressed instructions to the corresponding areas of the flash ROM.

Makinen, 4:18-32

See also Makinen, Figs. 2-4

8.6 updating the boot data list,	Makinen, as evidenced by the example citations below, discloses "updating the boot data list"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, updating the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claim 1.4.1 above.	

8.7 wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form.	Makinen, as evidenced by the example citations below, discloses "wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form"
To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, wherein the portion of the operating system is accessed and decompressed at a rate that is faster than accessing the loaded portion of the operating system from the first memory if the portion of the operating system was to be stored in the first memory in an uncompressed form), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions. Makinen discloses this limitation: *See* Claims 1-1.3 and 5.7 above. *See also* ***Look for additional references where accessing and decompressing occurs in less time than the time to load the boot data if the boot data were stored in an uncompressed form*** Compared to conventional ROM, flash ROM remains expensive. There therefore exists a desire to reduce the amount of flash ROM used in individual products. Furthermore, accessing flash ROM is relatively slow, significantly reducing the performance of electronic devices. Conventional computer architectures rely upon a complex instruction set computer (CISC) architecture. This utilises a large number (e.g. 1000) of instructions which define very specific tasks. The instructions are of variable length and are decoded by the computer's CPU before execution. There is described in Japanese non-examined patent publication no. 55-131848 a data processing unit which comprises a central processing unit, a main memory unit, and an external memory unit. A compressed program is stored in the external memory unit and, before commencing processing operations, the compressed program is read from the external memory unit in blocks and decompressed. The decompressed program is then written to the main memory unit. However, the compression ratio which can be achieved with CISC code is relatively low and it is not believed that the disclosure of JP-131848 has been widely used.	

Makinen, 1:34-54

Preferably, the method of the above first or second aspect of the invention comprises writing the decompressed instruction set to a random access memory (RAM). Thereafter, the decompressed instructions are read from the RAM by the CPU. It is noted that RAM typically offers high access speeds compared to slow (e.g. flash) ROM memory, giving a significant increase in system performance. In this case, increased speed also offers reduced power consumption compared to systems which use slow ROM memory and in which power is consumed even when the system is waiting to access the ROM.

Makinen, 2:40-50

Appendix C13
Invalidity of U.S. Patent 8,880,862 based on Makinen

9.1 The method of claim 8, further comprising: compressing an additional portion of the operating system that is not associated with the boot data list; and	Makinen, as evidenced by the example citations below, discloses "the method of claim 8, further comprising: compressing an additional portion of the operating system that is not associated with the boot data list"

To the extent that Realtime contends this reference does not explicitly disclose this element of this claim (for example, compressing an additional portion of the operating system that is not associated with the boot data list), Apple contends that one of skill in the art would understand the operation of booting a computer system to include the element that is missing similar to the manner in which the patentee relied upon such knowledge of skill in the art during prosecution. See Sections VI. and VII. of Apple's Invalidity Contentions.

Makinen discloses this limitation:

See Claim 1.1 above.

> A computer having a reduced instruction computer (RISC) architecture has a RISC central processing unit (CPU)(1) coupled to a RAM memory (3) and to a flash ROM memory (4). A set of compressed operating instructions (6,8), including a subset defining a compression method (8), are stored in the flash ROM (4) together with a set of uncompressed instructions (7) defining a compression algorithm. Upon booting of the computer, the uncompressed instructions (7) are read from the ROM (4) by the CPU (1) which then also reads the compressed instructions (6,8), decompresses them according to the decompression process (7), and writes the decompressed instructions (6′,8′) to the RAM (3). The compressed instructions (6,8) can be dynamically altered by the CPU (1), by generating an altered set of uncompressed instructions, compressing these in accordance with the now decompressed compression method (8′), and writing these to the flash ROM (4).

Makinen, Abstract

> At the heart of most modern electronic devices is a microprocessor or central processing unit which operates in accordance with a set of software operating instructions which together form an executable program. The instructions are stored in a digital memory which may be internal to the microprocessor or, as is more usually the case, externally connected to the microprocessor. The set of operating instructions generally define the basic input/output system (BIOS) of the microprocessor together with device drivers, libraries, and user applications.