S E C O N D   E D I T I O N

Computer Organization and Design

THE HARDWARE/SOFTWARE INTERFACE

John L. Hennessy
Stanford University

David A. Patterson
University of California, Berkeley

With a contribution by
James R. Larus
University of Wisconsin

Morgan Kaufmann Publishers, Inc.
San Francisco, California

INTEL - 1012

Sponsoring Editor Denise Penrose
Production Manager Yonie Overton
Production Editor Julie Pabst
Editorial Coordinator Jane Elliott
Text and Cover Design Ross Carron Design
Illustration Alexander Teshin Associates, with second edition modifications by Dartmouth Publishing, Inc.
Chapter Opener Illustrations Canary Studios
Copyeditor Ken DellaPenta
Composition Nancy Logan
Proofreader Jennifer McClain
Indexer Steve Rath
Printer Courier Corporation

Morgan Kaufmann Publishers, Inc.
Editorial and Sales Office:
340 Pine Street, Sixth Floor
San Francisco, CA 94104-3205
USA

Telephone 415/392-2665
Facsimile 415/982-2665
Email *mkp@mkp.com*
WWW *http://www.mkp.com*
Order toll free 800/745-7323

Advice, Praise, and Errors: Any correspondence related to this publication or intended for the authors should be sent electronically to *cod2bugs@mkp.com.* Information regarding error sightings is encouraged. Any error sightings that are accepted for correction in subsequent printings will be rewarded by the authors with a payment of $1.00 (U.S.) per correction at the time of their implementation in a reprint.

Library of Congress Cataloging-in-Publication Data
Patterson, David A.
Computer organization and design : the hardware/software interface / David A. Patterson, John L. Hennessy.—2nd ed.
p. cm.
Includes bibliographical references and index.
ISBN 1-55860-428-6 (cloth).—ISBN 1-55860-491-X (paper)
1. Computer organization. 2. Computers—Design and construction.
3. Computer interfaces. I. Hennessy, John L. II. Title
QA76.9.C643H46 1997
004.2'2—dc21 97-16050

INTEL - 1012

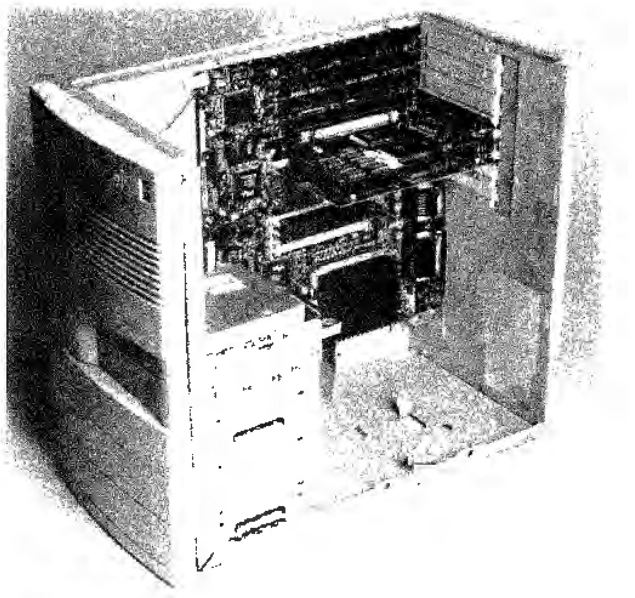

FIGURE 1.8 Inside a personal computer. The vertical board in the back is a printed circuit board (PC board), called the *motherboard* in a PC, that contains most of the electronics of the computer; Figure 1.11 is an overhead photograph of that board, rotated 90 degrees. The processor is the large black rectangle in the lower-right corner of the board. (Figure 1.9 is a photograph of the processor before it is placed in the black package.) The two large boards attached perpendicularly in the top third of the motherboard on the right contain input/output interfaces to the Ethernet local area network and a video card for a CRT. The two small boards attached perpendicularly to the middle of the motherboard contain the memory chips. The large box on the lower left is a cage for mounting additional drives; above it are a hard disk drive and a floppy disk drive. The power supply is normally to the right of this box, but it was removed to get a better view of the motherboard.

The *processor* is the active part of the board, following the instructions of a program to the letter. It adds numbers, tests numbers, signals I/O devices to activate, and so on. The processor is the large square below the memory boards in the lower-right corner of Figure 1.8. Occasionally, people call the processor the *CPU*, for the more bureaucratic-sounding *central processor unit*.

Descending even lower into the hardware, Figure 1.9 reveals details of the processor in Figure 1.8. The processor comprises two main components: datapath and control, the respective brawn and brain of the processor. The *datapath* performs the arithmetic operations, and *control* tells the datapath, memory, and I/O devices what to do according to the wishes of the instructions of the program. Chapter 5 explains the datapath and control for a straightforward implementation, and Chapter 6 describes the changes needed for a higher performance design.

The field is in the lower 16 bits of the word and we want 0s in the upper bits of the result of the `andi`. In general, a shift left of 32 – (n + m) followed by a shift right by 32 – n will isolate any n-bit field whose least significant bit is in bit m.

Since `addi` and `slti` are intended for signed numbers, it is not surprising that their immediate fields are sign-extended before use. Branch and data transfer address fields are sign-extended as well.

Perhaps it *is* surprising that `addiu` and `sltiu` also sign-extend their immediates, but they do. The `u` stands for unsigned, but in reality `addiu` is often used simply as an `add` instruction that cannot overflow, and hence we often want to add negative numbers. It's much harder to come up for an excuse that `sltiu` does not sign extend its immediate.

Since `andi` and `ori` normally work with unsigned integers, the immediates are treated as unsigned integers as well, meaning that they are expanded to 32 bits by padding with leading 0s instead of sign extension. Thus if the bit fields in the third line of the example above extended beyond the 16 least significant bits, the `andi` instruction would need a 32-bit constant to avoid clearing the upper portion of the fields.

The MIPS assembler creates 32-bit constants with the pair of instructions `lui` and `ori`; see Chapter 3, page 147 for an example of creating 32-bit constants using `lui` and `addi`.

4.5 Constructing an Arithmetic Logic Unit

*ALU n. [**A**rthritic **L**ogic **U**nit or (rare) **A**rithmetic **L**ogic **U**nit] A random-number generator supplied as standard with all computer systems.*

Stan Kelly-Bootle, *The Devil's DP Dictionary*, 1981

The *arithmetic logic unit* or *ALU* is the brawn of the computer, the device that performs the arithmetic operations like addition and subtraction or logical operations like AND and OR. This section constructs an ALU from the four hardware building blocks shown in Figure 4.8 (see Appendix B for more details on these building blocks). Cases 1, 2, and 4 in Figure 4.8 all have two inputs. We will sometimes use versions of these components with more than two inputs, confident that you can generalize from this simple example. In any case, Appendix B provides examples with more inputs. (You may wish to review sections B.1 through B.3 before proceeding further.)

Because the MIPS word is 32 bits wide, we need a 32-bit-wide ALU. Let's assume that we will connect 32 1-bit ALUs to create the desired ALU. We'll therefore start by constructing a 1-bit ALU.

A 1-Bit ALU

The logical operations are easiest, because they map directly onto the hardware components in Figure 4.8.

The Sum bit is set when exactly one input is 1 or when all three inputs are 1. The Sum results in a complex Boolean equation (recall that $\bar{a}$ means NOT a):

$$\text{Sum} = (a \cdot \bar{b} \cdot \overline{\text{CarryIn}}) + (\bar{a} \cdot b \cdot \overline{\text{CarryIn}}) + (\bar{a} \cdot \bar{b} \cdot \text{CarryIn}) + (a \cdot b \cdot \text{CarryIn})$$

The drawing of the logic for the Sum bit in the adder black box is left as an exercise (see Exercise 4.43).

Figure 4.14 shows a 1-bit ALU derived by combining the adder with the earlier components. Sometimes designers also want the ALU to perform a few more simple operations, such as generating 0. The easiest way to add an operation is to expand the multiplexor controlled by the Operation line and, for this example, to connect 0 directly to the new input of that expanded multiplexor.

A 32-Bit ALU

Now that we have completed the 1-bit ALU, the full 32-bit ALU is created by connecting adjacent "black boxes." Using *xi* to mean the *i*th bit of *x*, Figure 4.15 shows a 32-bit ALU. Just as a single stone can cause ripples to radiate to the shores of a quiet lake, a single carry out of the least significant bit (Result0) can ripple all the way through the adder, causing a carry out of the most significant bit (Result31). Hence, the adder created by directly linking the carries of 1-bit adders is called a *ripple carry* adder. We'll see a faster way to connect the 1-bit adders starting on page 241.

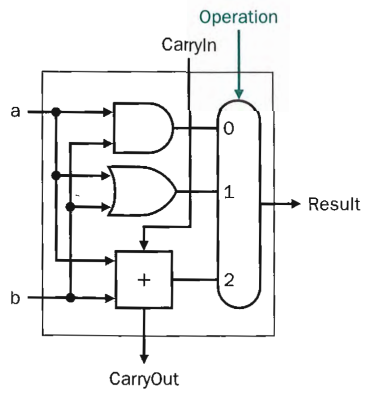

FIGURE 4.14 A 1-bit ALU that performs AND, OR, and addition (see Figure 4.13).

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | sales@docketalarm.com | 1-866-77-FASTCASE

fastcase® Smarter legal research.