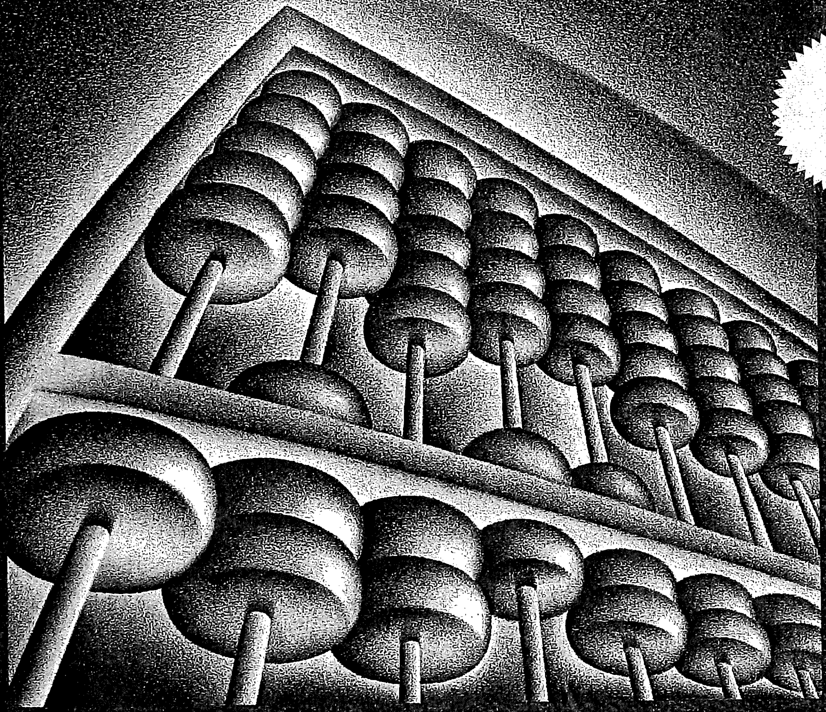
ELSEVIER

COMPUTER ORGANIZATION AND DESIGN

THE HARDWARE/SOFTWARE INTERFACE



DAVID A. PATTERSON
JOHN L. HENNESSY

MORGAN KAUFMANN

Apple, Inc. et al. v.
Memory Integrity, LLC
IPR2015-00159. -00161,
-00163, -00172
EXHIBIT
Memory Integrity - 2005

THIRD EDITION

Computer Organization and Design

THE HARDWARE/SOFTWARE INTERFACE

David A. Patterson
University of California, Berkeley

John L. Hennessy
Stanford University

With a contribution by
Peter J. Ashenden
Ashenden Designs Pty Ltd

James R. Larus
Microsoft Research

Daniel J. Sorin
Duke University

ELSEVIER

AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO

Morgan Kaufmann is an imprint of Elsevier

MK®
MORGAN KAUFMANN PUBLISHERS

Senior Editor	Denise E. M. Penrose
Publishing Services Manager	Simon Crump
Editorial Assistant	Summer Block
Cover Design	Ross Caron Design
Cover and Chapter Illustration	Chris Asimoudis
Text Design	GGS Book Services
Composition	Nancy Logan and Dartmouth Publishing, Inc.
Technical Illustration	Dartmouth Publishing, Inc.
Copyeditor	Ken DellaPenta
Proofreader	Jacqui Brownstein
Indexer	Linda Buskus
Interior printer	Courier
Cover printer	Courier

Morgan Kaufmann Publishers is an imprint of Elsevier.
500 Sansome Street, Suite 400, San Francisco, CA 94111

This book is printed on acid-free paper.

Library of Congress Cataloging-in-Publication Data
Application submitted

ISBN: 1-55860-604-1

For information on all Morgan Kaufmann publications,
visit our Web site at www.mkp.com.

Printed in the United States of America
04 05 06 07 08 5 4 3 2 1

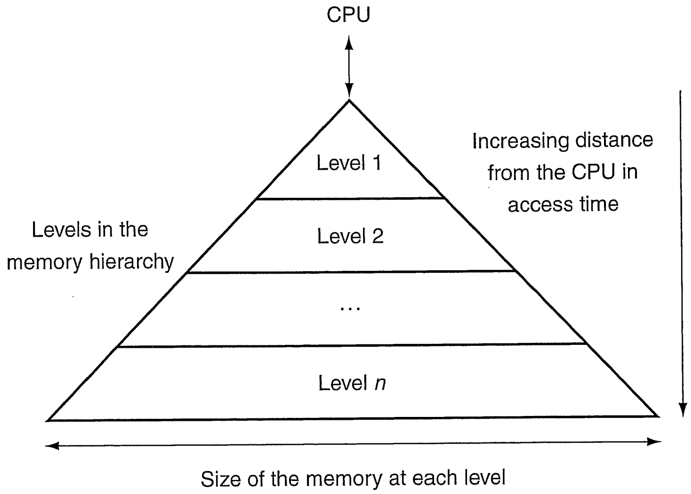


FIGURE 7.3 This diagram shows the structure of a memory hierarchy: as the distance from the processor increases, so does the size. This structure with the appropriate operating mechanisms allows the processor to have an access time that is determined primarily by level 1 of the hierarchy and yet have a memory as large as level n. Maintaining this illusion is the subject of this chapter. Although the local disk is normally the bottom of the hierarchy, some systems use tape or a file server over a local area network as the next levels of the hierarchy.

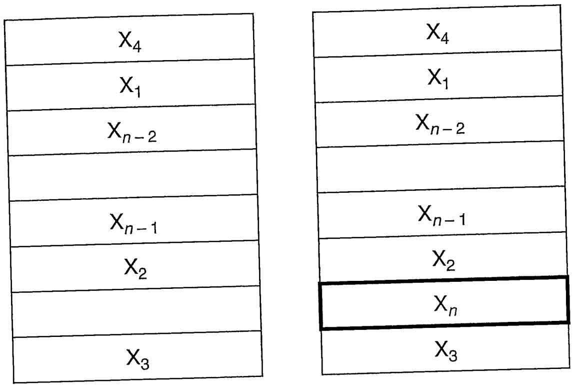
7.2 The Basics of Caches

Cache: a safe place for hiding or storing things.

Webster's New World Dictionary of the American Language, Third College Edition (1988)

In our library example, the desk acted as a cache—a safe place to store things (books) that we needed to examine. *Cache* was the name chosen to represent the level of the memory hierarchy between the processor and main memory in the first commercial computer to have this extra level. Today, although this remains the dominant use of the word *cache*, the term is also used to refer to any storage managed to take advantage of locality of access. Caches first appeared in research computers in the early 1960s and in production computers later in that same decade; every general-purpose computer built today, from servers to low-power embedded processors, includes caches.

In this section, we begin by looking at a very simple cache in which the processor requests are each one word and the blocks also consist of a single word. (Readers already familiar with cache basics may want to skip to Section 7.3 on page 492.)



a. Before the reference to X_n b. After the reference to X_n

FIGURE 7.4 The cache just before and just after a reference to a word X_n that is not initially in the cache. This reference causes a miss that forces the cache to fetch X_n from memory and insert it into the cache.

Figure 7.4 shows such a simple cache, before and after requesting a data item that is not initially in the cache. Before the request, the cache contains a collection of recent references $X_1, X_2, \ldots, X_{n-1}$, and the processor requests a word X_n that is not in the cache. This request results in a miss, and the word X_n is brought from memory into cache.

direct-mapped cache A cache structure in which each memory location is mapped to exactly one location in the cache.

In looking at the scenario in Figure 7.4, there are two questions to answer: How do we know if a data item is in the cache? Moreover, if it is, how do we find it? The answers to these two questions are related. If each word can go in exactly one place in the cache, then it is straightforward to find the word if it is in the cache. The simplest way to assign a location in the cache for each word in memory is to assign the cache location based on the *address* of the word in memory. This cache structure is called **direct mapped**, since each memory location is mapped directly to exactly one location in the cache. The typical mapping between addresses and cache locations for a direct-mapped cache is usually simple. For example, almost all direct-mapped caches use the mapping

(Block address) modulo (Number of cache blocks in the cache)

This mapping is attractive because if the number of entries in the cache is a power of two, then modulo can be computed simply by using the low-order $\log_2$ (cache size in blocks) bits of the address; hence the cache may be accessed directly with the low-order bits. For example, Figure 7.5 shows how the memory addresses

DOCKET ALARM

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.