

CONCISE ENCYCLOPEDIA OF COMPUTER SCIENCE

Editor:

Edwin D. Reilly



WILEY

SEL
QA76.15
C654
2004

Copyright © 2004 John Wiley & Sons, Ltd,
The Atrium,
Southern Gate,
Chichester,
West Sussex,
PO19 8SQ, England

Telephone (+44) 1243 779777
Email (for orders and customer service enquires): cs-books@wiley.co.uk
Visit our Home Page on www.wiley.co.uk or www.wiley.com

All Rights Reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except under the terms of the Copyright, Designs and Patents Act 1988 or under the terms of a licence issued by the Copyright Licensing Agency Ltd, 90 Tottenham Court Road, London, W1P 0LP, UK, without the permission in writing of the Publisher. Requests to the Publisher should be addressed to the Permissions Department, John Wiley & Sons, Ltd, The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England, or e-mailed to permreq@wiley.co.uk, or faxed to (+44) 1243 770620.

This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold on the understanding that the Publisher is not engaged in rendering professional services. If professional advice or other expert assistance is required, the services of a competent professional should be sought.

Other Wiley Editorial Offices

John Wiley & Sons, Inc., 111 River Street,
Hoboken, NJ 07030, USA

Jossey-Bass, 989 Market Street,
San Francisco, CA 94103-1741, USA

Wiley-VCH Verlag GmbH, Boschstr. 12,
D-69469 Weinheim, Germany

John Wiley & Sons Australia, Ltd, 33 Park Road,
Milton, Queensland, 4064, Australia

John Wiley & Sons (Asia) Pte Ltd, 2 Clementi Loop #02-01,
Jin Xing Distripark, Singapore 129809

John Wiley & Sons Canada Ltd, 22 Worcester Road,
Etobicoke, Ontario, Canada, M9W 1L1

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

Library of Congress Cataloguing-in-Publication Data

Concise Encyclopedia of Computer Science/Editor, Edwin D. Reilly.

p. cm.

Includes bibliographical references and index.

ISBN 0-470-09095-2 (pbk.)

1. Computer science—Encyclopedias. I. Reilly, Edwin D.

QA76.15.C654 2004

004'.03—dc22

2004009421

British Library Cataloguing in Publication Data

A catalogue record for this book is available from the British Library

ISBN 0-470-09095-2

Typeset in 9.5/12.5pt AGaramond by Laserwords Private Limited, Chennai, India.

Printed and bound in Great Britain by Biddles Ltd, King's Lynn.

This book is printed on acid-free paper responsibly manufactured from sustainable forestry in which at least two trees are planted for each one used for paper production.

CACHE MEMORY

For articles on related subjects, see

- + Associative Memory
- + Buffer
- + Computer Architecture
- + Memory
- + Memory Hierarchy
- + Virtual Memory

Introduction

A *cache memory* is a small, high-speed buffer used for temporary storage of those portions of the contents of some larger memory that are (believed to be) currently in use. The most common use of a cache memory is in the CPU of a computer system, where it holds or buffers the contents of main memory. Cache memories can also be used to hold the contents of the disk (a disk cache) or mass storage (e.g. a tape cache).

A cache memory is normally both significantly faster and significantly smaller than the memory whose contents it caches. Like any component of the memory hierarchy, it is useful only if it can satisfy a large fraction of the references to the larger memory. In practice, caches are very effective in a wide variety of situations because of the *principle of locality* (see WORKING SET), which is an empirical observation that most of the time the information in use is either the same information that was recently in use (temporal locality), or is information "nearby" the information recently used (spatial locality). Thus, a cache typically operates by retaining copies of blocks of storage, each containing recently used information.

Caches are usually transparent or invisible to the processor. The CPU generates a main memory address when it wishes to read or write data. When a cache is added to this design, it is interposed between the CPU and the main memory. The cache thus receives the main memory address, and determines, through some sort of associative search using the main memory address as the key, whether it already holds the data corresponding to that address. If so, in the case of a read, it replies to the CPU with the data, and the main memory is not

referenced. When the data sought is retrieved from the cache, there is a cache *hit*, otherwise a cache *miss*.

Cache Operation

A CPU (central processing unit) typically consists of three major components: the *instruction unit*, which fetches and decodes instructions (see INSTRUCTION DECODING); the *execution unit*, which executes the instructions; and the *storage unit*, which contains the cache, the TLB (translation lookaside buffer), and the translator. The operation of the cache is inseparable from the operation of the rest of the storage unit, and we discuss the cache in that context.

Figure 1 is a diagram showing the CPU and main memory. Figure 2 presents a very simple design for a CPU cache memory, showing that the cache consists of a number of entries. Each entry consists of an address tag, a valid bit, and a *line* or *block* of data. If the valid bit is set to 1, the data is a copy of the data held in the main memory locations identified by the address tag. If the valid bit is set to 0, then the corresponding data field does not currently hold valid data as a result of changes to the contents of main memory. When the instruction (I) or execution (E) units of the CPU generate a main memory address, that address is presented to the cache, which associatively and simultaneously compares the address with the address field of each entry for which the valid bit is on. If a match is found (a *hit*), on a read, the cache then replies to the I- or E-unit with the requested information. If there is no match, then the cache fetches the line containing the desired information from main memory, extracts the target information and

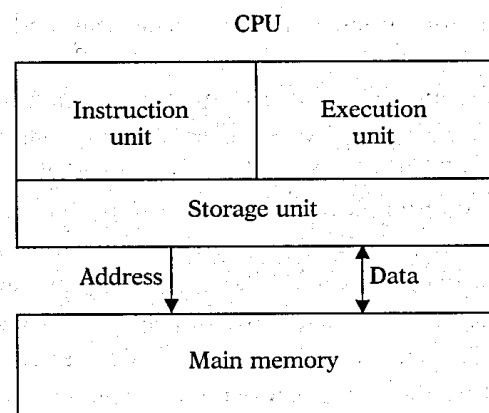


Figure 1. A CPU and main memory.

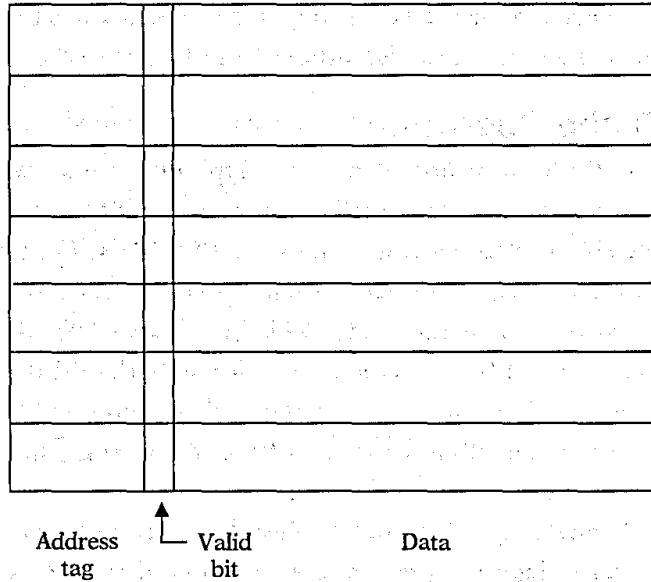


Figure 2. Elements of a simple cache memory.

sends it to the I- or E-unit, and also stores the line in a cache entry, replacing the previous contents of that entry. In the case of a write, the data in the cache is updated and, concurrently, the write is also transmitted to main memory, where the main memory copy of the data is updated.

Typically, CPU cache memories will experience 95–99.9% hits and TLBs generally will have 98–99.9% hits, depending on the cache size, line size, and workload.

Other Types of Cache

Caches can be used in numerous ways in computer systems. A common type of cache is the *disk cache*. The disk cache holds portions of the material on the disk in some sort of faster storage, usually semiconductor storage. The disk cache can be located within the disk enclosure, in which case it is managed by the (single) disk controller in a file server or multidisk controller, or the CPU can do the disk caching itself by using a portion of main memory for that purpose. Disk caches are reasonably effective, with typical read hit ratios in the range of 70–90%. Issues in disk cache design are similar to those for CPU caches; the designer must consider the cache size, the cache location (CPU, server, disk controller), the write policy (write-through, copy-back), replacement algorithm, fetch algorithm, and block size. A particularly important consideration in disk cache

design is the problem of reliability in the case of system failure or power loss. In general, an operating system assumes that data written to disk is completely safe and immune to system failure. If data “written” to disk is actually in the disk cache, this assumption is violated, with potentially severe consequences for system integrity. Either write-through or battery backup is typically used for disk caches in systems that are expected to be highly reliable.

Database systems almost always do some amount of caching; typically, they maintain a region in main memory, which is used to hold recently referenced blocks of data. This is similar in concept and operation to disk caching.

Caching is used within operating systems to avoid lengthy and redundant computations. For example, the directory structure is used to translate file names to file descriptors, and frequently the same name is translated repeatedly. By caching recently performed translations, use of the directory structure can be avoided.

Caching is used in various ways in accessing the World Wide Web (*q.v.*). A Web browser caches recently used pages locally at the user’s computer. Typically, a Web server will also cache recently referenced pages, so that it isn’t necessary to retrieve popular pages from disk each time they are referenced.

Bibliography

- 1982. Smith, A. J. “Cache Memories,” *ACM Computing Surveys*, 14(3), 473–530.
- 1997. Uhlig, R. A., and Mudge, T. N. “Trace-driven Memory Simulation: A Survey,” *ACM Computing Surveys*, 29(2), 128–170.
- 1999. Peir, J.-K., Hsu, W., and Smith, A. J. “Implementation Issues in Modern Cache Memories,” *IEEE Transactions on Computers*, 48(2), 100–110.

Alan Jay Smith

CAD/CAM

See COMPUTER-AIDED DESIGN/COMPUTER-AIDED MANUFACTURING (CAD/CAM).