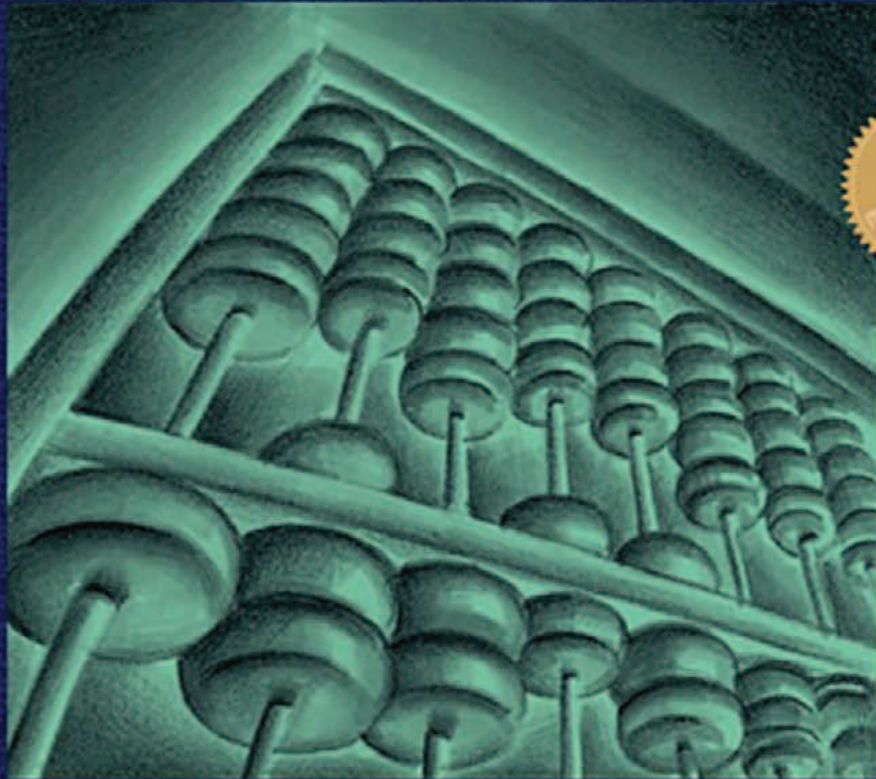




# COMPUTER ORGANIZATION AND DESIGN

THE HARDWARE/SOFTWARE INTERFACE



THIRD  
3  
EDITION

DAVID A. PATTERSON  
JOHN L. HENNESSY



T H I R D   E D I T I O N

# **Computer Organization Design**

T H E   H A R D W A R E / S O F T W A R E   I N T E R F A C E

## ACKNOWLEDGEMENTS

Figures 1.9, 1.15 Courtesy of Intel.

Figure 1.11 Courtesy of Storage Technology Corp.

Figures 1.7.1, 1.7.2, 6.13.2 Courtesy of the Charles Babbage Institute, University of Minnesota Libraries, Minneapolis.

Figures 1.7.3, 6.13.1, 6.13.3, 7.9.3, 8.11.2 Courtesy of IBM.

Figure 1.7.4 Courtesy of Cray Inc.

Figure 1.7.5 Courtesy of Apple Computer, Inc.

Figure 1.7.6 Courtesy of the Computer History Museum.

Figure 7.33 Courtesy of AMD.

Figures 7.9.1, 7.9.2 Courtesy of Museum of Science, Boston.

Figure 7.9.4 Courtesy of MIPS Technologies, Inc.

Figure 8.3 ©Peg Skorpinski.

Figure 8.11.1 Courtesy of the Computer Museum of America.

Figure 8.11.3 Courtesy of the Commercial Computing Museum.

Figures 9.11.2, 9.11.3 Courtesy of NASA Ames Research Center.

Figure 9.11.4 Courtesy of Lawrence Livermore National Laboratory.

### Computers in the Real World:

Photo of “A Laotian villager,” courtesy of David Sanger.

Photo of an “Indian villager,” property of Encore Software, Ltd., India.

Photos of “Block and students” and “a pop-up archival satellite tag,” courtesy of Professor Barbara Block. Photos by Scott Taylor.

Photos of “Professor Dawson and student” and “the Mica micromote,” courtesy of AP/World Wide Photos.

Photos of “images of pottery fragments” and “a computer reconstruction,” courtesy of Andrew Willis and David B. Cooper, Brown University, Division of Engineering.

Photo of “the Eurostar TGV train,” by Jos van der Kolk.

Photo of “the interior of a Eurostar TGV cab,” by Andy Veitch.

Photo of “firefighter Ken Whitten,” courtesy of World Economic Forum.

Graphic of an “artificial retina,” © The San Francisco Chronicle. Reprinted by permission.

Image of “A laser scan of Michelangelo’s statue of David,” courtesy of Marc Levoy and Dr. Franca Falletti, director of the Galleria dell’Accademia, Italy.

“An image from the Sistine Chapel,” courtesy of Luca Pezzati. IR image recorded using the scanner for IR reflectography of the INOA (National Institute for Applied Optics, <http://arte.ino.it>) at the Opificio delle Pietre Dure in Florence.

T H I R D   E D I T I O N

# 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



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	GGG 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.

© 2005 by Elsevier Inc. All rights reserved.

Designations used by companies to distinguish their products are often claimed as trademarks or registered trademarks. In all instances in which Morgan Kaufmann Publishers is aware of a claim, the product names appear in initial capital or all capital letters. Readers, however, should contact the appropriate companies for more complete information regarding trademarks and registration.

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, scanning, or otherwise—without prior written permission of the publisher.

Permissions may be sought directly from Elsevier's Science & Technology Rights Department in Oxford, UK: phone: (+44) 1865 843830, fax: (+44) 1865 853333, e-mail: [permissions@elsevier.com.uk](mailto:permissions@elsevier.com.uk). You may also complete your request on-line via the Elsevier homepage (<http://elsevier.com>) by selecting "Customer Support" and then "Obtaining Permissions."

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](http://www.mkp.com).


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

# Contents

Preface ix


## CHAPTERS



### **1 Computer Abstractions and Technology 2**

- 1.1 Introduction 3
- 1.2 Below Your Program 11
- 1.3 Under the Covers 15
- 1.4 Real Stuff: Manufacturing Pentium 4 Chips 28
- 1.5 Fallacies and Pitfalls 33
- 1.6 Concluding Remarks 35
-  1.7 Historical Perspective and Further Reading 36
- 1.8 Exercises 36

### **COMPUTERS IN THE REAL WORLD** **Information Technology for the 4 Billion without IT 44**

### **2 Instructions: Language of the Computer 46**


- 2.1 Introduction 48
- 2.2 Operations of the Computer Hardware 49
- 2.3 Operands of the Computer Hardware 52
- 2.4 Representing Instructions in the Computer 60
- 2.5 Logical Operations 68
- 2.6 Instructions for Making Decisions 72
- 2.7 Supporting Procedures in Computer Hardware 79
- 2.8 Communicating with People 90
- 2.9 MIPS Addressing for 32-Bit Immediates and Addresses 95
- 2.10 Translating and Starting a Program 106
- 2.11 How Compilers Optimize 116
-  2.12 How Compilers Work: An Introduction 121

- 2.13 A C Sort Example to Put It All Together 121
-  2.14 Implementing an Object-Oriented Language 130
- 2.15 Arrays versus Pointers 130
- 2.16 Real Stuff: IA-32 Instructions 134
- 2.17 Fallacies and Pitfalls 143
- 2.18 Concluding Remarks 145
-  2.19 Historical Perspective and Further Reading 147
- 2.20 Exercises 147

## COMPUTERS IN THE REAL WORLD

### Helping Save Our Environment with Data 156


## 3 Arithmetic for Computers 158

- 3.1 Introduction 160
- 3.2 Signed and Unsigned Numbers 160
- 3.3 Addition and Subtraction 170
- 3.4 Multiplication 176
- 3.5 Division 183
- 3.6 Floating Point 189
- 3.7 Real Stuff: Floating Point in the IA-32 217
- 3.8 Fallacies and Pitfalls 220
- 3.9 Concluding Remarks 225
-  3.10 Historical Perspective and Further Reading 229
- 3.11 Exercises 229

## COMPUTERS IN THE REAL WORLD

### Reconstructing the Ancient World 236




## 4 Assessing and Understanding Performance 238

- 4.1 Introduction 240
- 4.2 CPU Performance and Its Factors 246
- 4.3 Evaluating Performance 254
- 4.4 Real Stuff: Two SPEC Benchmarks and the Performance of Recent Intel Processors 259
- 4.5 Fallacies and Pitfalls 266
- 4.6 Concluding Remarks 270
-  4.7 Historical Perspective and Further Reading 272
- 4.8 Exercises 272



## COMPUTERS IN THE REAL WORLD

### Moving People Faster and More Safely 280

**5****The Processor: Datapath and Control 282**

- 5.1 Introduction 284
- 5.2 Logic Design Conventions 289
- 5.3 Building a Datapath 292
- 5.4 A Simple Implementation Scheme 300
- 5.5 A Multicycle Implementation 318
- 5.6 Exceptions 340
-  5.7 Microprogramming: Simplifying Control Design 346
-  5.8 An Introduction to Digital Design Using a Hardware Design Language 346
- 5.9 Real Stuff: The Organization of Recent Pentium Implementations 347
- 5.10 Fallacies and Pitfalls 350
- 5.11 Concluding Remarks 352
-  5.12 Historical Perspective and Further Reading 353
- 5.13 Exercises 354


**COMPUTERS IN THE REAL WORLD****Empowering the Disabled 366****6****Enhancing Performance with Pipelining 368**

- 6.1 An Overview of Pipelining 370
- 6.2 A Pipelined Datapath 384
- 6.3 Pipelined Control 399
- 6.4 Data Hazards and Forwarding 402
- 6.5 Data Hazards and Stalls 413
- 6.6 Branch Hazards 416
-  6.7 Using a Hardware Description Language to Describe and Model a Pipeline 426
- 6.8 Exceptions 427
- 6.9 Advanced Pipelining: Extracting More Performance 432
- 6.10 Real Stuff: The Pentium 4 Pipeline 448
- 6.11 Fallacies and Pitfalls 451
- 6.12 Concluding Remarks 452
-  6.13 Historical Perspective and Further Reading 454
- 6.14 Exercises 454



**COMPUTERS IN THE REAL WORLD****Mass Communication without Gatekeepers 464**



**7****Large and Fast: Exploiting Memory Hierarchy 466**

- 7.1 Introduction 468
- 7.2 The Basics of Caches 473
- 7.3 Measuring and Improving Cache Performance 492
- 7.4 Virtual Memory 511
- 7.5 A Common Framework for Memory Hierarchies 538
- 7.6 Real Stuff: The Pentium P4 and the AMD Opteron Memory Hierarchies 546
- 7.7 Fallacies and Pitfalls 550
- 7.8 Concluding Remarks 552
-  7.9 Historical Perspective and Further Reading 555
- 7.10 Exercises 555

**COMPUTERS IN THE REAL WORLD****Saving the World's Art Treasures 562****8****Storage, Networks, and Other Peripherals 564**

- 8.1 Introduction 566
- 8.2 Disk Storage and Dependability 569
-  8.3 Networks 580
- 8.4 Buses and Other Connections between Processors, Memory, and I/O Devices 581
- 8.5 Interfacing I/O Devices to the Processor, Memory, and Operating System 588
- 8.6 I/O Performance Measures: Examples from Disk and File Systems 597
- 8.7 Designing an I/O System 600
- 8.8 Real Stuff: A Digital Camera 603
- 8.9 Fallacies and Pitfalls 606
- 8.10 Concluding Remarks 609
-  8.11 Historical Perspective and Further Reading 611
- 8.12 Exercises 611

**COMPUTERS IN THE REAL WORLD****Saving Lives through Better Diagnosis 622****9****Multiprocessors and Clusters 9-2**

- 9.1 Introduction 9-4
- 9.2 Programming Multiprocessors 9-8
- 9.3 Multiprocessors Connected by a Single Bus 9-11

- 9.4 Multiprocessors Connected by a Network 9-20
- 9.5 Clusters 9-25
- 9.6 Network Topologies 9-27
- 9.7 Multiprocessors Inside a Chip and Multithreading 9-30
- 9.8 Real Stuff: The Google Cluster of PCs 9-34
- 9.9 Fallacies and Pitfalls 9-39
- 9.10 Concluding Remarks 9-42
- 9.11 Historical Perspective and Further Reading 9-47
- 9.12 Exercises 9-55

## A P P E N D I C E S



### **Assemblers, Linkers, and the SPIM Simulator A-2**

- A.1 Introduction A-3
- A.2 Assemblers A-10
- A.3 Linkers A-18
- A.4 Loading A-19
- A.5 Memory Usage A-20
- A.6 Procedure Call Convention A-22
- A.7 Exceptions and Interrupts A-33
- A.8 Input and Output A-38
- A.9 SPIM A-40
- A.10 MIPS R2000 Assembly Language A-45
- A.11 Concluding Remarks A-81
- A.12 Exercises A-82



### **The Basics of Logic Design B-2**

- B.1 Introduction B-3
- B.2 Gates, Truth Tables, and Logic Equations B-4
- B.3 Combinational Logic B-8
- B.4 Using a Hardware Description Language B-20
- B.5 Constructing a Basic Arithmetic Logic Unit B-26
- B.6 Faster Addition: Carry Lookahead B-38
- B.7 Clocks B-47
- B.8 Memory Elements: Flip-flops, Latches, and Registers B-49
- B.9 Memory Elements: SRAMs and DRAMs B-57
- B.10 Finite State Machines B-67
- B.11 Timing Methodologies B-72

- B.12 Field Programmable Devices B-77
- B.13 Concluding Remarks B-78
- B.14 Exercises B-79



## Mapping Control to Hardware C-2

- C.1 Introduction C-3
- C.2 Implementing Combinational Control Units C-4
- C.3 Implementing Finite State Machine Control C-8
- C.4 Implementing the Next-State Function with a Sequencer C-21
- C.5 Translating a Microprogram to Hardware C-27
- C.6 Concluding Remarks C-31
- C.7 Exercises C-32



## A Survey of RISC Architectures for Desktop, Server, and Embedded Computers D-2

- D.1 Introduction D-3
- D.2 Addressing Modes and Instruction Formats D-5
- D.3 Instructions: The MIPS Core Subset D-9
- D.4 Instructions: Multimedia Extensions of the Desktop/Server RISCs D-16
- D.5 Instructions: Digital Signal-Processing Extensions of the Embedded RISCs D-19
- D.6 Instructions: Common Extensions to MIPS Core D-20
- D.7 Instructions Unique to MIPS64 D-25
- D.8 Instructions Unique to Alpha D-27
- D.9 Instructions Unique to SPARC v.9 D-29
- D.10 Instructions Unique to PowerPC D-32
- D.11 Instructions Unique to PA-RISC 2.0 D-34
- D.12 Instructions Unique to ARM D-36
- D.13 Instructions Unique to Thumb D-38
- D.14 Instructions Unique to SuperH D-39
- D.15 Instructions Unique to M32R D-40
- D.16 Instructions Unique to MIPS16 D-41
- D.17 Concluding Remarks D-43
- D.18 Acknowledgments D-46
- D.19 References D-47

Index I-1

-  Glossary G-1
-  Further Reading FR-1

# Preface

*The most beautiful thing we can experience is the mysterious.  
It is the source of all true art and science.*

**Albert Einstein, *What I Believe*, 1930**

## About This Book

We believe that learning in computer science and engineering should reflect the current state of the field, as well as introduce the principles that are shaping computing. We also feel that readers in every specialty of computing need to appreciate the organizational paradigms that determine the capabilities, performance, and, ultimately, the success of computer systems.

Modern computer technology requires professionals of every computing specialty to understand both hardware and software. The interaction between hardware and software at a variety of levels also offers a framework for understanding the fundamentals of computing. Whether your primary interest is hardware or software, computer science or electrical engineering, the central ideas in computer organization and design are the same. Thus, our emphasis in this book is to show the relationship between hardware and software and to focus on the concepts that are the basis for current computers.

The audience for this book includes those with little experience in assembly language or logic design who need to understand basic computer organization as well as readers with backgrounds in assembly language and/or logic design who want to learn how to design a computer or understand how a system works and why it performs as it does.

## About the Other Book

Some readers may be familiar with *Computer Architecture: A Quantitative Approach*, popularly known as Hennessy and Patterson. (This book in turn is called Patterson and Hennessy.) Our motivation in writing that book was to describe the principles of computer architecture using solid engineering funda-

This Page Intentionally Left Blank

The goal of this book is to explore performance trade-offs. We used an approach that combined examples and measurements, based on commercial systems, to create realistic design experiences. Our goal was to demonstrate that computer architecture could be learned using quantitative methodologies instead of a descriptive approach. It is intended for the serious computing professional who wants a detailed understanding of computers.

A majority of the readers for this book do not plan to become computer architects. The performance of future software systems will be dramatically affected, however, by how well software designers understand the basic hardware techniques at work in a system. Thus, compiler writers, operating system designers, database programmers, and most other software engineers need a firm grounding in the principles presented in this book. Similarly, hardware designers must understand clearly the effects of their work on software applications.

Thus, we knew that this book had to be much more than a subset of the material in *Computer Architecture*, and the material was extensively revised to match the different audience. We were so happy with the result that the subsequent editions of *Computer Architecture* were revised to remove most of the introductory material; hence, there is much less overlap today than with the first editions of both books.

### **Changes for the Third Edition**

We had six major goals for the third edition of *Computer Organization and Design*: make the book work equally well for readers with a software focus or with a hardware focus; improve pedagogy in general; enhance understanding of program performance; update the technical content to reflect changes in the industry since the publication of the second edition in 1998; tie the ideas from the book more closely to the real world *outside* the computing industry; and reduce the size of this book.

First, the table on the next page shows the hardware and software paths through the material. Chapters 1, 4, and 7 are found on both paths, no matter what the experience or the focus. Chapters 2 and 3 are likely to be review material for the hardware-oriented, but are essential reading for the software-oriented, especially for those readers interested in learning more about compilers and object-oriented programming languages. The first sections of Chapters 5 and 6 give overviews for those with a software focus. Those with a hardware focus, however, will find that these chapters present core material; they may also, depending on background, want to read Appendix B on logic design first and the sections on microprogramming and how to use hardware description languages to specify control. Chapter 8 on input/output is key to readers with a software focus and should be read if time permits by others. The last chapter on multiprocessors and clusters is again a question of time for the reader. Even the history sections show this balanced focus; they include short histories of programming languages, compilers, numerical software, operating systems, networking protocols, and databases.

<i>Chapter or Appendix</i>	<i>Sections</i>	<i>Software Focus</i>	<i>Hardware Focus</i>
1. Computer Abstractions and Technology	1.1 to 1.6		
	1.7 (History)		
2. Instructions: Language of the Computer	2.1 to 2.11		
	2.12 (Compilers)		
	2.13 (C sort)		
	2.14 (Java)		
	2.15 to 2.18		
	2.19 (History)		
3. Arithmetic for Computers	3.1 to 3.11		
	3.12 (History)		
D. RISC instruction set architectures	D.1 to D.19		
4. Assessing and Understanding Performance	4.1 to 4.6		
	4.7 (History)		
B. The Basics of Logic Design	B.1 to B.13		
5. The Processor: Datapath and Control	5.1 (Overview)		
	5.2 to 5.7		
	5.8 (Microcode)		
	5.9 (Verilog)		
	5.10 to 5.12		
	5.13 (History)		
C. Mapping Control to Hardware	C.1 to C.6		
6. Enhancing Performance with Pipelining	6.1 (Overview)		
	6.2 to 6.6		
	6.7 (verilog)		
	6.8 to 6.9		
	6.10 to 6.12		
	6.13 (History)		
7. Large and Fast: Exploiting Memory Hierarchy	7.1 to 7.8		
	7.9 (History)		
8. Storage, Networks, and Other Peripherals	8.1 to 8.2		
	8.3 (Networks)		
	8.4 to 8.10		
	8.13 (History)		
9. Multiprocessors and Clusters	9.1 to 9.10		
	9.11 (History)		
A. Assemblers, Linkers, and the SPIM Simulator	A.1 to A.12		
Computers in the Real World	Between Chapters		

Read carefully      Read if have time      Reference   
 Review or read      Read for culture

The next goal was to improve the exposition of the ideas in the book, based on difficulties mentioned by readers of the second edition. We added five new book elements to help. To make the book work better as a reference, we placed definitions of new terms in the margins at their first occurrence. We hope this will help readers find the sections when they want to refer back to material they have already read. Another change was the insertion of the “Check Yourself” sections, which we added to help readers to check their comprehension of the material on the first time through it. A third change is that added extra exercises in the “For More Practice” section. Fourth, we added the answers to the “Check Yourself” sections and to the For More Practice exercises to help readers see for themselves if they understand the material by comparing their answers to the book. The final new book element was inspired by the “Green Card” of the IBM System/360. We believe that you will find that the MIPS Reference Data Card will be a handy reference when writing MIPS assembly language programs. Our idea is that you will remove the card from the front of the book, fold it in half, and keep it in your pocket, just as IBM S/360 programmers did in the 1960s.

Third, computers are so complex today that understanding the performance of a program involves understanding a good deal about the underlying principles and the organization of a given computer. Our goal is that readers of this book should be able to understand the performance of their programs and how to improve it. To aid in that goal, we added a new book element called “Understanding Program Performance” in several chapters. These sections often give concrete examples of how ideas in the chapter affect performance of real programs.

Fourth, in the interval since the second edition of this book, Moore’s law has marched onward so that we now have processors with 200 million transistors, DRAM chips with a billion transistors, and clock rates of multiple gigahertz. The “Real Stuff” examples have been updated to describe such chips. This edition also includes AMD64/IA-32e, the 64-bit address version of the long-lived 80x86 architecture, which appears to be the nemesis of the more recent IA-64. It also reflects the transition from parallel buses to serial networks and switches. Later chapters describe Google, which was born after the second edition, in terms of its cluster technology and in novel uses of search.

Fifth, although many computer science and engineering students enjoy information technology for technology’s sake, some have more altruistic interests. This latter group tends to have more women and underrepresented minorities. Consequently, we have added a new book element, “Computers in the Real World,” two-page layouts found between each chapter. Our perspective is that information technology is more valuable for humanity than most other topics you could study—whether it is preserving our art heritage, helping the Third World, saving our environment, or even changing political systems—and so we demonstrate our view with concrete examples of nontraditional applications. We think readers of these segments will have a greater appreciation of the computing culture beyond

the inherently interesting technology, much like those who read the history sections at the end of each chapter

Finally, books are like people: they usually get larger as they get older. By using technology, we have managed to do all the above and yet shrink the page count by hundreds of pages. As the table illustrates, the core portion of the book for hardware and software readers is on paper, but sections that some readers would value more than others are found on the companion CD. This technology also allows your authors to provide longer histories and more extensive exercises without concerns about lengthening the book. Once we added the CD to the book, we could then include a great deal of free software and tutorials that many instructors have told us they would like to use in their courses. This hybrid paper-CD publication weighs about 30% less than it did six years ago—an impressive goal for books as well as for people.

### **Instructor Support**

We have collected a great deal of material to help instructors teach courses using this book. Solutions to exercises, figures from the book, lecture notes, lecture slides, and other materials are available to adopters from the publisher. Check the publisher's Web site for more information:

[www.mkp.com/companions/1558606041](http://www.mkp.com/companions/1558606041)

### **Concluding Remarks**

If you read the following acknowledgments section, you will see that we went to great lengths to correct mistakes. Since a book goes through many printings, we have the opportunity to make even more corrections. If you uncover any remaining, resilient bugs, please contact the publisher by electronic mail at [cod3bugs@mkp.com](mailto:cod3bugs@mkp.com) or by low-tech mail using the address found on the copyright page. The first person to report a technical error will be awarded a \$1.00 bounty upon its implementation in future printings of the book!

This book is truly collaborative, despite one of us running a major university. Together we brainstormed about the ideas and method of presentation, then individually wrote about one-half of the chapters and acted as reviewer for every draft of the other half. The page count suggests we again wrote almost exactly the same number of pages. Thus, we equally share the blame for what you are about to read.

### **Acknowledgments for the Third Edition**

We'd like to again express our appreciation to **Jim Larus** for his willingness in contributing his expertise on assembly language programming, as well as for welcoming readers of this book to use the simulator he developed and maintains. Our



exercise editor **Dan Sorin** took on the Herculean task of adding new exercises and answers. **Peter Ashenden** worked similarly hard to collect and organize the companion CD.

We are grateful to the many instructors who answered the publisher's surveys, reviewed our proposals, and attended focus groups to analyze and respond to our plans for this edition. They include the following individuals: Michael Anderson (University of Hartford), David Bader (University of New Mexico), Rusty Baldwin (Air Force Institute of Technology), John Barr (Ithaca College), Jack Briner (Charleston Southern University), Mats Brorsson (KTH, Sweden), Colin Brown (Franklin University), Lori Carter (Point Loma Nazarene University), John Casey (Northeastern University), Gene Chase (Messiah College), George Cheney (University of Massachusetts, Lowell), Daniel Citron (Jerusalem College of Technology, Israel), Albert Cohen (INRIA, France), Lloyd Dickman (PathScale), Jose Duato (Universidad Politécnic de Valencia, Spain), Ben Dugan (University of Washington), Derek Eager (University of Saskatchewan, Canada), Magnus Ekman (Chalmers University of Technology, Sweden), Ata Elahi (Southern Connecticut State University), Soundararajan Ezekiel (Indiana University of Pennsylvania), Ernest Ferguson (Northwest Missouri State University), Michael Fry (Lebanon Valley College, Pennsylvania), R. Gaede (University of Arkansas at Little Rock), Jean-Luc Gaudiot (University of California, Irvine), Thomas Gendreau (University of Wisconsin, La Crosse), George Georgiou (California State University, San Bernardino), Paul Gillard (Memorial University of Newfoundland, Canada), Joe Grimes (California Polytechnic State University, SLO), Max Hailperin (Gustavus Adolphus College), Jayantha Herath (St. Cloud State University, Minnesota), Mark Hill (University of Wisconsin, Madison), Michael Hsaio (Virginia Tech), Richard Hughey (University of California, Santa Cruz), Tony Jebara (Columbia University), Elizabeth Johnson (Xavier University), Peter Kogge (University of Notre Dame), Morris Lancaster (BAH), Doug Lawrence (University of Montana), David Lilja (University of Minnesota), Nam Ling (Santa Clara University, California), Paul Lum (Agilent Technologies), Stephen Mann (University of Waterloo, Canada), Diana Marculescu (Carnegie Mellon University), Margaret McMahon (U.S. Naval Academy Computer Science), Uwe Meyer-Baese (Florida State University), Chris Milner (University of Virginia), Tom Pittman (Southwest Baptist University), Jalel Rejeb (San Jose State University, California), Bill Siever (University of Missouri, Rolla), Kevin Skadron (University of Virginia), Pam Smallwood (Regis University, Colorado), K. Stuart Smith (Rocky Mountain College), William J. Taffe (Plymouth State University), Michael E. Thomodakis (Texas A&M University), Ruppa K. Thulasiram (University of Manitoba, Canada), Ye Tung (University of South Alabama), Steve VanderLeest (Calvin College), Neal R. Wagner (University of Texas at San Antonio), and Kent Wilken (University of California, Davis).

We are grateful too to those who carefully read our draft manuscripts; some read successive drafts to help ensure new errors didn't creep in as we revised. They include Krste Asanovic (Massachusetts Institute of Technology), Jean-Loup Baer (University of Washington), David Brooks (Harvard University), Doug Clark (Princeton University), Dan Connors (University of Colorado at Boulder), Matt Farrens (University of California, Davis), Manoj Franklin (University of Maryland College Park), John Greiner (Rice University), David Harris (Harvey Mudd College), Paul Hilfinger (University of California, Berkeley), Norm Jouppi (Hewlett-Packard), David Kaeli (Northeastern University), David Oppenheimer (University of California, Berkeley), Timothy Pinkston (University of Southern California), Mark Smotherman (Clemson University), and David Wood (University of Wisconsin, Madison).

To help us meet our goal of creating 70% new exercises and solutions for this edition, we recruited several graduate students recommended to us by their professors. We are grateful for their creativity and persistence: Michael Black (University of Maryland), Lei Chen (University of Rochester), Nirav Dave (Massachusetts Institute of Technology), Wael El Essawy (University of Rochester), Nikil Mehta (Brown University), Nicholas Nelson (University of Rochester), Aaron Smith (University of Texas, Austin), and Charlie Wang (Duke University).

We would like to especially thank **Mark Smotherman** for making a careful final pass to find technical and writing glitches that significantly improved the quality of this edition.

We wish to thank the extended Morgan Kaufmann family for agreeing to publish this book again under the able leadership of **Denise Penrose**. She developed the vision of the hybrid paper-CD book and recruited the many people above who played important roles in developing the book.

**Simon Crump** managed the book production process, and **Summer Block** coordinated the surveying of users and their responses. We thank also the many freelance vendors who contributed to this volume, especially **Nancy Logan** and **Dartmouth Publishing, Inc.**, our compositors.

The contributions of the nearly 100 people we mentioned here have made this third edition our best book yet. Enjoy!

**David A. Patterson**

**John L. Hennessy**