

经 典 原 版 书 库

计算机组成与设计

硬件/软件接口

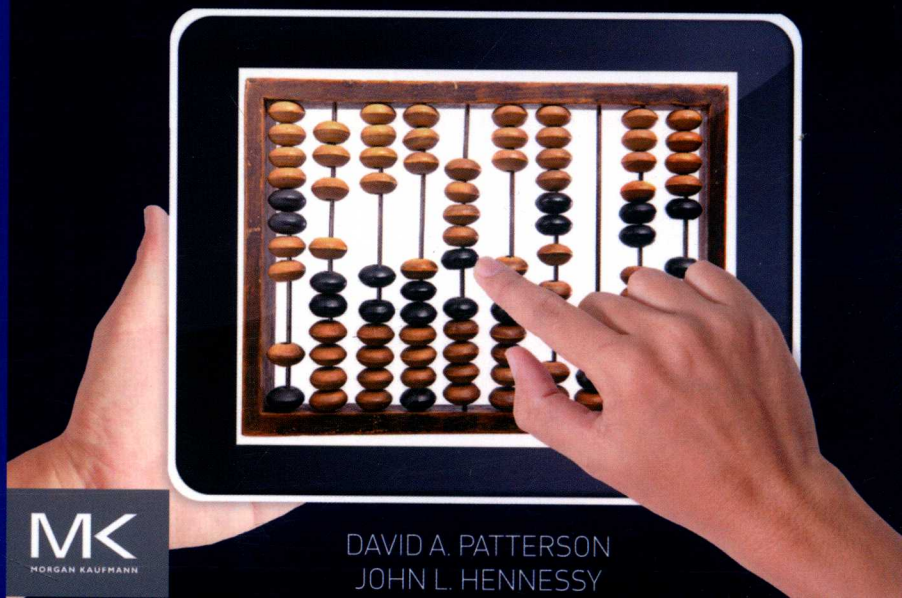
[美] 戴维·A. 帕特森 约翰·L. 亨尼斯 著
David A. Patterson John L. Hennessy

(英文版·原书第5版·RISC-V版)

COMPUTER ORGANIZATION AND DESIGN

THE HARDWARE/SOFTWARE INTERFACE

RISC-V EDITION



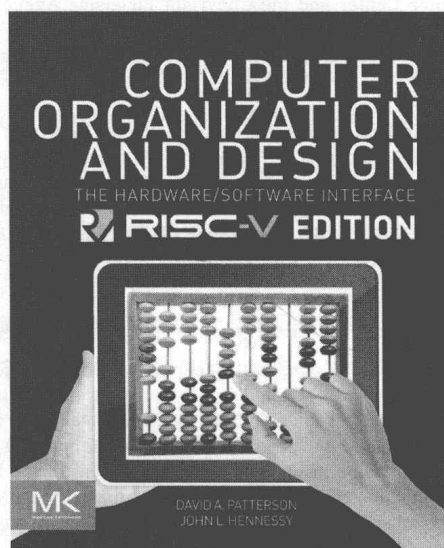
经 典 原 版 书 库

计算机组成与设计

硬件/软件接口

(英文版·原书第5版·RISC-V版)

Computer Organization and Design
The Hardware/Software Interface, RISC-V Edition



[美] 戴维·A. 帕特森 约翰·L. 亨尼斯 著
David A. Patterson John L. Hennessy



机械工业出版社
China Machine Press

图书在版编目 (CIP) 数据

计算机组成与设计: 硬件 / 软件接口 (英文版 · 原书第 5 版 · RISC-V 版)/(美) 戴维 · A. 帕特森 (David A. Patterson), (美) 约翰 · L. 亨尼斯 (John L. Hennessy) 著. —北京: 机械工业出版社, 2019.7 (经典原版书库)

书名原文: Computer Organization and Design: The Hardware/Software Interface, RISC-V Edition

ISBN 978-7-111-63111-8

I. 计… II. ①戴… ②约… III. ①计算机体系结构-英文 ②微型计算机-接口设备-英文
IV. ①TP303 ②TP364

中国版本图书馆 CIP 数据核字 (2019) 第 130603 号

本书版权登记号: 图字 01-2019-3870

Computer Organization and Design: The Hardware/Software Interface, RISC-V Edition

David A. Patterson, John L. Hennessy

ISBN: 9780128122754

Copyright © 2018 Elsevier Inc. All rights reserved.

Authorized Chinese translation published by China Machine Press.

计算机组成与设计: 硬件 / 软件接口 (英文版 · 原书第 5 版 · RISC-V 版)

ISBN: 9787111631118

Copyright © Elsevier Inc. and China Machine Press. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or any information storage and retrieval system, without permission in writing from Elsevier (Singapore) Pte Ltd. Details on how to seek permission, further information about the Elsevier's permissions policies and arrangements with organizations such as the Copyright Clearance Center and the Copyright Licensing Agency, can be found at our website: www.elsevier.com/permissions.

This book and the individual contributions contained in it are protected under copyright by Elsevier Inc. and China Machine Press (other than as may be noted herein).

Online resources are not available with this reprint.

This edition of Computer Organization and Design: The Hardware/Software Interface, RISC-V Edition is published by China Machine Press under arrangement with ELSEVIER INC.

This edition is authorized for sale in China only, excluding Hong Kong, Macau and Taiwan. Unauthorized export of this edition is a violation of the Copyright Act. Violation of this Law is subject to Civil and Criminal Penalties.

本版由 ELSEVIER INC. 授权机械工业出版社在中国大陆地区 (不包括香港、澳门以及台湾地区) 出版发行。

本版仅限在中国大陆地区 (不包括香港、澳门以及台湾地区) 出版及标价销售。未经许可之出口, 视为违反著作权法, 将受民事及刑事法律之制裁。

本书封底贴有 Elsevier 防伪标签, 无标签者不得销售。

Notice

Knowledge and best practice in this field are constantly changing. As new research and experience broaden our understanding, changes in research methods, professional practices, or medical treatment may become necessary. Practitioners and researchers must always rely on their own experience and knowledge in evaluating and using any information, methods, compounds or experiments described herein. Because of rapid advances in the medical sciences, in particular, independent verification of diagnoses and drug dosages should be made. To the fullest extent of the law, no responsibility is assumed by Elsevier, authors, editors or contributors in relation to the adaptation or for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions, or ideas contained in the material herein.

出版发行: 机械工业出版社 (北京市西城区百万庄大街 22 号 邮政编码: 100037)

责任编辑: 曲 熠

责任校对: 殷 虹

印 刷: 北京诚信伟业印刷有限公司

版 次: 2019 年 7 月第 1 版第 1 次印刷

开 本: 186mm × 240mm 1/16

印 张: 43.25

书 号: ISBN 978-7-111-63111-8

定 价: 229.00 元

凡购本书, 如有缺页、倒页、脱页, 由本社发行部调换

客服热线: (010) 88378991 88379833

投稿热线: (010) 88379604

购书热线: (010) 68326294

读者信箱: hzjsj@hzbook.com

版权所有 · 侵权必究

封底无防伪标均为盗版

本书法律顾问: 北京大成律师事务所 韩光 / 邹晓东

出版者的话

文艺复兴以来，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的优势，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭示了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短的现状下，美国等发达国家在其计算机科学发展的几十年间积淀和发展的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起到积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章公司较早意识到“出版要为教育服务”。自1998年开始，我们就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与Pearson、McGraw-Hill、Elsevier、MIT、John Wiley & Sons、Cengage等世界著名出版公司建立了良好的合作关系，从它们现有的数百种教材中甄选出Andrew S. Tanenbaum、Bjarne Stroustrup、Brian W. Kernighan、Dennis Ritchie、Jim Gray、Afred V. Aho、John E. Hopcroft、Jeffrey D. Ullman、Abraham Silberschatz、William Stallings、Donald E. Knuth、John L. Hennessy、Larry L. Peterson等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及珍藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力相助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专门为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近500个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证。随着计算机科学与技术专业学科建设的不断完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都将步入一个新的阶段，我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方式如下：

华章网站：www.hzbook.com

电子邮件：hzjsj@hzbook.com

联系电话：(010) 88379604

联系地址：北京市西城区百万庄南街1号

邮政编码：100037



华章科技图书出版中心

In Praise of *Computer Organization and Design: The Hardware/Software Interface*

“Textbook selection is often a frustrating act of compromise—pedagogy, content coverage, quality of exposition, level of rigor, cost. *Computer Organization and Design* is the rare book that hits all the right notes across the board, without compromise. It is not only the premier computer organization textbook, it is a shining example of what all computer science textbooks could and should be.”

—Michael Goldweber, *Xavier University*

“I have been using *Computer Organization and Design* for years, from the very first edition. This new edition is yet another outstanding improvement on an already classic text. The evolution from desktop computing to mobile computing to Big Data brings new coverage of embedded processors such as the ARM, new material on how software and hardware interact to increase performance, and cloud computing. All this without sacrificing the fundamentals.”

—Ed Harcourt, *St. Lawrence University*

“To Millennials: *Computer Organization and Design* is the computer architecture book you should keep on your (virtual) bookshelf. The book is both old and new, because it develops venerable principles—Moore’s Law, abstraction, common case fast, redundancy, memory hierarchies, parallelism, and pipelining—but illustrates them with contemporary designs.”

—Mark D. Hill, *University of Wisconsin-Madison*

“The new edition of *Computer Organization and Design* keeps pace with advances in emerging embedded and many-core (GPU) systems, where tablets and smartphones will/are quickly becoming our new desktops. This text acknowledges these changes, but continues to provide a rich foundation of the fundamentals in computer organization and design which will be needed for the designers of hardware and software that power this new class of devices and systems.”

—Dave Kaeli, *Northeastern University*

“*Computer Organization and Design* provides more than an introduction to computer architecture. It prepares the reader for the changes necessary to meet the ever-increasing performance needs of mobile systems and big data processing at a time that difficulties in semiconductor scaling are making all systems power constrained. In this new era for computing, hardware and software must be co-designed and system-level architecture is as critical as component-level optimizations.”

—Christos Kozyrakis, *Stanford University*

“Patterson and Hennessy brilliantly address the issues in ever-changing computer hardware architectures, emphasizing on interactions among hardware and software components at various abstraction levels. By interspersing I/O and parallelism concepts with a variety of mechanisms in hardware and software throughout the book, the new edition achieves an excellent holistic presentation of computer architecture for the post-PC era. This book is an essential guide to hardware and software professionals facing energy efficiency and parallelization challenges in Tablet PC to Cloud computing.”

—Jae C. Oh, *Syracuse University*

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, energy, 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 recent switch from uniprocessor to multicore microprocessors confirmed the soundness of this perspective, given since the first edition. While programmers could ignore the advice and rely on computer architects, compiler writers, and silicon engineers to make their programs run faster or be more energy-efficient without change, that era is over. For programs to run faster, they must become parallel. While the goal of many researchers is to make it possible for programmers to be unaware of the underlying parallel nature of the hardware they are programming, it will take many years to realize this vision. Our view is that for at least the next decade, most programmers are going to have to understand the hardware/software interface if they want programs to run efficiently on parallel 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 often called Patterson and Hennessy.) Our motivation in writing the earlier book was to describe the principles of computer architecture using solid engineering fundamentals and quantitative cost/performance tradeoffs. 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 was intended for the serious computing professional who wanted a detailed understanding of computers.

A majority of the readers for this book do not plan to become computer architects. The performance and energy efficiency 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.

Why RISC-V for This Edition?

The choice of instruction set architecture is clearly critical to the pedagogy of a computer architecture textbook. We didn't want an instruction set that required describing unnecessary baroque features for someone's first instruction set, no matter how popular it is. Ideally, your initial instruction set should be an exemplar, just like your first love. Surprisingly, you remember both fondly.

Since there were so many choices at the time, for the first edition of *Computer Architecture: A Quantitative Approach* we invented our own RISC-style instruction set. Given the growing popularity and the simple elegance of the MIPS instruction set, we switched to it for the first edition of this book and to later editions of the other book. MIPS has served us and our readers well.

It's been 20 years since we made that switch, and while billions of chips that use MIPS continue to be shipped, they are typically found embedded devices where the instruction set is nearly invisible. Thus, for a while now it's been hard to find a real computer on which readers can download and run MIPS programs.

The good news is that an open instruction set that adheres closely to the RISC principles has recently debuted, and it is rapidly gaining a following. RISC-V, which was developed originally at UC Berkeley, not only cleans up the quirks of the MIPS

instruction set, but it offers a simple, elegant, modern take on what instruction sets should look like in 2017.

Moreover, because it is not proprietary, there are open-source RISC-V simulators, compilers, debuggers, and so on easily available and even open-source RISC-V implementations available written in hardware description languages. In addition, there will soon be low-cost hardware platforms on which to run RISC-V programs. Readers will not only benefit from studying these RISC-V designs, they will be able to modify them and go through the implementation process in order to understand the impact of their hypothetical changes on performance, die size, and energy.

This is an exciting opportunity for the computing industry as well as for education, and thus at the time of this writing more than 40 companies have joined the RISC-V foundation. This sponsor list includes virtually all the major players except for ARM and Intel, including AMD, Google, Hewlett Packard Enterprise, IBM, Microsoft, NVIDIA, Oracle, and Qualcomm.

It is for these reasons that we wrote a RISC-V edition of this book, and we are switching *Computer Architecture: A Quantitative Approach* to RISC-V as well.

Given that RISC-V offers both 32-bit address instructions and 64-bit address instructions with essentially the same instruction set, we could have switched instruction sets but kept the address size at 32 bits. Our publisher polled the faculty who used the book and found that 75% either preferred larger addresses or were neutral, so we increased the address space to 64 bits, which may make more sense today than 32 bits.

The only changes for the RISC-V edition from the MIPS edition are those associated with the change in instruction sets, which primarily affects Chapter 2, Chapter 3, the virtual memory section in Chapter 5, and the short VMIPS example in Chapter 6. In Chapter 4, we switched to RISC-V instructions, changed several figures, and added a few “Elaboration” sections, but the changes were simpler than we had feared. Chapter 1 and the rest of the appendices are virtually unchanged. The extensive online documentation and combined with the magnitude of RISC-V make it difficult to come up with a replacement for the MIPS version of Appendix A (“Assemblers, Linkers, and the SPIM Simulator” in the MIPS Fifth Edition). Instead, Chapters 2, 3, and 5 include quick overviews of the hundreds of RISC-V instructions outside of the core RISC-V instructions that we cover in detail in the rest of the book.









































































Note that we are not (yet) saying that we are permanently switching to RISC-V. For example, in addition to this new RISC-V edition, there are ARMv8 and MIPS versions available for sale now. One possibility is that there will be a demand for all versions for future editions of the book, or for just one. We’ll cross that bridge when we come to it. For now, we look forward to your reaction to and feedback on this effort.

Changes for the Fifth Edition

We had six major goals for the fifth edition of *Computer Organization and Design* demonstrate the importance of understanding hardware with a running example; highlight main themes across the topics using margin icons that are introduced

early; update examples to reflect changeover from PC era to post-PC era; spread the material on I/O throughout the book rather than isolating it into a single chapter; update the technical content to reflect changes in the industry since the publication of the fourth edition in 2009; and put appendices and optional sections online instead of including a CD to lower costs and to make this edition viable as an electronic book.

Before discussing the goals in detail, let's look at the table on the next page. It shows the hardware and software paths through the material. Chapters 1, 4, 5, and 6 are found on both paths, no matter what the experience or the focus. Chapter 1 discusses the importance of energy and how it motivates the switch from single core to multicore microprocessors and introduces the eight great ideas in computer architecture. Chapter 2 is likely to be review material for the hardware-oriented, but it is essential reading for the software-oriented, especially for those readers interested in learning more about compilers and object-oriented programming languages. Chapter 3 is for readers interested in constructing a datapath or in learning more about floating-point arithmetic. Some will skip parts of Chapter 3, either because they don't need them, or because they offer a review. However, we introduce the running example of matrix multiply in this chapter, showing how subword parallels offers a fourfold improvement, so don't skip Sections 3.6 to 3.8. Chapter 4 explains pipelined processors. Sections 4.1, 4.5, and 4.10 give overviews, and Section 4.12 gives the next performance boost for matrix multiply for those with a software focus. Those with a hardware focus, however, will find that this chapter presents core material; they may also, depending on their background, want to read Appendix A on logic design first. The last chapter, on multicores, multiprocessors, and clusters, is mostly new content and should be read by everyone. It was significantly reorganized in this edition to make the flow of ideas more natural and to include much more depth on GPUs, warehouse-scale computers, and the hardware–software interface of network interface cards that are key to clusters.

Chapter or Appendix	Sections	Software focus	Hardware focus
1. Computer Abstractions and Technology	1.1 to 1.11		
	 1.12 (History)		
2. Instructions: Language of the Computer	2.1 to 2.14		
	 2.15 (Compilers & Java)		
	2.16 to 2.20		
	 2.21 (History)		
D. RISC Instruction-Set Architectures	 D.1 to D.17		
3. Arithmetic for Computers	3.1 to 3.5		
	3.6 to 3.8 (Subword Parallelism)		
	3.9 to 3.10 (Fallacies)		
	 3.11 (History)		
A. The Basics of Logic Design	A.1 to A.13		
4. The Processor	4.1 (Overview)		
	4.2 (Logic Conventions)		
	4.3 to 4.4 (Simple Implementation)		
	4.5 (Pipelining Overview)		
	4.6 (Pipelined Datapath)		
	4.7 to 4.9 (Hazards, Exceptions)		
	4.10 to 4.12 (Parallel, Real Stuff)		
	 4.13 (Verilog Pipeline Control)		
	4.14 to 4.15 (Fallacies)		
	 4.16 (History)		
C. Mapping Control to Hardware	 C.1 to C.6		
5. Large and Fast: Exploiting Memory Hierarchy	5.1 to 5.10		
	 5.11 (Redundant Arrays of Inexpensive Disks)		
	 5.12 (Verilog Cache Controller)		
	5.13 to 5.17		
	 5.18 (History)		
6. Parallel Process from Client to Cloud	6.1 to 6.8		
	 6.9 (Networks)		
	6.10 to 6.14		
	 6.15 (History)		
B. Graphics Processor Units	 B.1 to B.13		

Read carefully



Read if have time



Reference



Review or read



Read for culture



The first of the six goals for this fifth edition was to demonstrate the importance of understanding modern hardware to get good performance and energy efficiency with a concrete example. As mentioned above, we start with subword parallelism in Chapter 3 to improve matrix multiply by a factor of 4. We double performance in Chapter 4 by unrolling the loop to demonstrate the value of instruction-level parallelism. Chapter 5 doubles performance again by optimizing for caches using blocking. Finally, Chapter 6 demonstrates a speedup of 14 from 16 processors by using thread-level parallelism. All four optimizations in total add just 24 lines of C code to our initial matrix multiply example.

The second goal was to help readers separate the forest from the trees by identifying eight great ideas of computer architecture early and then pointing out all the places they occur throughout the rest of the book. We use (hopefully) easy-to-remember margin icons and highlight the corresponding word in the text to remind readers of these eight themes. There are nearly 100 citations in the book. No chapter has less than seven examples of great ideas, and no idea is cited less than five times. Performance via parallelism, pipelining, and prediction are the three most popular great ideas, followed closely by Moore's Law. Chapter 4, The Processor, is the one with the most examples, which is not a surprise since it probably received the most attention from computer architects. The one great idea found in every chapter is performance via parallelism, which is a pleasant observation given the recent emphasis in parallelism in the field and in editions of this book.

The third goal was to recognize the generation change in computing from the PC era to the post-PC era by this edition with our examples and material. Thus, Chapter 1 dives into the guts of a tablet computer rather than a PC, and Chapter 6 describes the computing infrastructure of the cloud. We also feature the ARM, which is the instruction set of choice in the personal mobile devices of the post-PC era, as well as the x86 instruction set that dominated the PC era and (so far) dominates cloud computing.

The fourth goal was to spread the I/O material throughout the book rather than have it in its own chapter, much as we spread parallelism throughout all the chapters in the fourth edition. Hence, I/O material in this edition can be found in Sections 1.4, 4.9, 5.2, 5.5, 5.11, and 6.9. The thought is that readers (and instructors) are more likely to cover I/O if it's not segregated to its own chapter.

This is a fast-moving field, and, as is always the case for our new editions, an important goal is to update the technical content. The running example is the ARM Cortex A53 and the Intel Core i7, reflecting our post-PC era. Other highlights include a tutorial on GPUs that explains their unique terminology, more depth on the warehouse-scale computers that make up the cloud, and a deep dive into 10 Gigabyte Ethernet cards.

To keep the main book short and compatible with electronic books, we placed the optional material as online appendices instead of on a companion CD as in prior editions.

Finally, we updated all the exercises in the book.

While some elements changed, we have preserved useful book elements from prior editions. To make the book work better as a reference, we still place definitions of new terms in the margins at their first occurrence. The book element called

“Understanding Program Performance” sections helps readers understand the performance of their programs and how to improve it, just as the “Hardware/Software Interface” book element helped readers understand the tradeoffs at this interface. “The Big Picture” section remains so that the reader sees the forest despite all the trees. “Check Yourself” sections help readers to confirm their comprehension of the material on the first time through with answers provided at the end of each chapter. This edition still includes the green RISC-V reference card, which was inspired by the “Green Card” of the IBM System/360. This card has been updated and should be a handy reference when writing RISC-V assembly language programs.

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 slides, and other materials are available to instructors who register with the publisher. In addition, the companion Web site provides links to a free RISC-V software. Check the publisher’s Web site for more information:

textbooks.elsevier.com/9780128122754

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 codRISCVbugs@mkp.com or by low-tech mail using the address found on the copyright page.

This edition is the third break in the long-standing collaboration between Hennessy and Patterson, which started in 1989. The demands of running one of the world’s great universities meant that President Hennessy could no longer make the substantial commitment to create a new edition. The remaining author felt once again like a tightrope walker without a safety net. Hence, the people in the acknowledgments and Berkeley colleagues played an even larger role in shaping the contents of this book. Nevertheless, this time around there is only one author to blame for the new material in what you are about to read.

Acknowledgments

With every edition of this book, we are very fortunate to receive help from many readers, reviewers, and contributors. Each of these people has helped to make this book better.

We are grateful for the assistance of **Khaled Benkrid** and his colleagues at ARM Ltd., who carefully reviewed the ARM-related material and provided helpful feedback.

Chapter 6 was so extensively revised that we did a separate review for ideas and contents, and I made changes based on the feedback from every reviewer. I’d like to thank **Christos Kozyrakis** of Stanford University for suggesting using the network

interface for clusters to demonstrate the hardware–software interface of I/O and for suggestions on organizing the rest of the chapter; **Mario Flagsilk** of Stanford University for providing details, diagrams, and performance measurements of the NetFPGA NIC; and the following for suggestions on how to improve the chapter: **David Kaeli** of Northeastern University, **Partha Ranganathan** of HP Labs, **David Wood** of the University of Wisconsin, and my Berkeley colleagues **Siamak Faridani**, **Shoaib Kamil**, **Yunsup Lee**, **Zhangxi Tan**, and **Andrew Waterman**.

Special thanks goes to **Rimas Avizenis** of UC Berkeley, who developed the various versions of matrix multiply and supplied the performance numbers as well. As I worked with his father while I was a graduate student at UCLA, it was a nice symmetry to work with Rimas at UCB.

I also wish to thank my longtime collaborator **Randy Katz** of UC Berkeley, who helped develop the concept of great ideas in computer architecture as part of the extensive revision of an undergraduate class that we did together.

I'd like to thank **David Kirk**, **John Nickolls**, and their colleagues at NVIDIA (Michael Garland, John Montrym, Doug Voorhies, Lars Nyland, Erik Lindholm, Paulius Micikevicius, Massimiliano Fatica, Stuart Oberman, and Vasily Volkov) for writing the first in-depth appendix on GPUs. I'd like to express again my appreciation to **Jim Larus**, recently named Dean of the School of Computer and Communications Science at EPFL, for his willingness in contributing his expertise on assembly language programming, as well as for welcoming readers of this book with regard to using the simulator he developed and maintains.

I am also very grateful to **Zachary Kurmas** of Grand Valley State University, who updated and created new exercises, based on originals created by **Perry Alexander** (The University of Kansas); **Jason Bakos** (University of South Carolina); **Javier Bruguera** (Universidade de Santiago de Compostela); **Matthew Farrens** (University of California, Davis); **David Kaeli** (Northeastern University); **Nicole Kaiyan** (University of Adelaide); **John Oliver** (Cal Poly, San Luis Obispo); **Milos Prvulovic** (Georgia Tech); **Jichuan Chang** (Google); **Jacob Leverich** (Stanford); **Kevin Lim** (Hewlett-Packard); and **Partha Ranganathan** (Google).

Additional thanks goes to **Peter Ashenden** for updating the lecture slides.

I am grateful to the many instructors who have answered the publisher's surveys, reviewed our proposals, and attended focus groups. They include the following individuals: Focus Groups: Bruce Barton (Suffolk County Community College), Jeff Braun (Montana Tech), Ed Gehringer (North Carolina State), Michael Goldweber (Xavier University), Ed Harcourt (St. Lawrence University), Mark Hill (University of Wisconsin, Madison), Patrick Homer (University of Arizona), Norm Jouppi (HP Labs), Dave Kaeli (Northeastern University), Christos Kozyrakis (Stanford University), Jae C. Oh (Syracuse University), Lu Peng (LSU), Milos Prvulovic (Georgia Tech), Partha Ranganathan (HP Labs), David Wood (University of Wisconsin), Craig Zilles (University of Illinois at Urbana-Champaign). Surveys and Reviews: Mahmoud Abou-Nasr (Wayne State University), Perry Alexander (The University of Kansas), Behnam Arad (Sacramento State University), Hakan Aydin (George Mason University), Hussein Badr (State University of New York at Stony Brook), Mac Baker (Virginia Military Institute), Ron Barnes (George Mason University),

Douglas Blough (Georgia Institute of Technology), Kevin Bolding (Seattle Pacific University), Miodrag Bolic (University of Ottawa), John Bonomo (Westminster College), Jeff Braun (Montana Tech), Tom Briggs (Shippensburg University), Mike Bright (Grove City College), Scott Burgess (Humboldt State University), Fazli Can (Bilkent University), Warren R. Carithers (Rochester Institute of Technology), Bruce Carlton (Mesa Community College), Nicholas Carter (University of Illinois at Urbana-Champaign), Anthony Cocchi (The City University of New York), Don Cooley (Utah State University), Gene Cooperman (Northeastern University), Robert D. Cupper (Allegheny College), Amy Csizmar Dalal (Carleton College), Daniel Dalle (Université de Sherbrooke), Edward W. Davis (North Carolina State University), Nathaniel J. Davis (Air Force Institute of Technology), Molisa Derk (Oklahoma City University), Andrea Di Blas (Stanford University), Derek Eager (University of Saskatchewan), Ata Elahi (Southern Connecticut State University), Ernest Ferguson (Northwest Missouri State University), Rhonda Kay Gaede (The University of Alabama), Etienne M. Gagnon (L'Université du Québec à Montréal), Costa Gerousis (Christopher Newport University), Paul Gillard (Memorial University of Newfoundland), Michael Goldweber (Xavier University), Georgia Grant (College of San Mateo), Paul V. Gratz (Texas A&M University), Merrill Hall (The Master's College), Tyson Hall (Southern Adventist University), Ed Harcourt (St. Lawrence University), Justin E. Harlow (University of South Florida), Paul F. Hemler (Hampden-Sydney College), Jayantha Herath (St. Cloud State University), Martin Herbordt (Boston University), Steve J. Hodges (Cabrillo College), Kenneth Hopkinson (Cornell University), Bill Hsu (San Francisco State University), Dalton Hunkins (St. Bonaventure University), Baback Izadi (State University of New York—New Paltz), Reza Jafari, Robert W. Johnson (Colorado Technical University), Bharat Joshi (University of North Carolina, Charlotte), Nagarajan Kandasamy (Drexel University), Rajiv Kapadia, Ryan Kastner (University of California, Santa Barbara), E.J. Kim (Texas A&M University), Jihong Kim (Seoul National University), Jim Kirk (Union University), Geoffrey S. Knauth (Lycoming College), Manish M. Kochhal (Wayne State), Suzan Koknar-Tezel (Saint Joseph's University), Angkul Kongmunvattana (Columbus State University), April Kontostathis (Ursinus College), Christos Kozyrakis (Stanford University), Danny Krizanc (Wesleyan University), Ashok Kumar, S. Kumar (The University of Texas), Zachary Kurmas (Grand Valley State University), Adrian Lauf (University of Louisville), Robert N. Lea (University of Houston), Alvin Lebeck (Duke University), Baoxin Li (Arizona State University), Li Liao (University of Delaware), Gary Livingston (University of Massachusetts), Michael Lyle, Douglas W. Lynn (Oregon Institute of Technology), Yashwant K Malaiya (Colorado State University), Stephen Mann (University of Waterloo), Bill Mark (University of Texas at Austin), Ananda Mondal (Clafin University), Alvin Moser (Seattle University),

Walid Najjar (University of California, Riverside), Vijaykrishnan Narayanan (Penn State University), Danial J. Neebel (Loras College), Victor Nelson (Auburn University), John Nestor (Lafayette College), Jae C. Oh (Syracuse University), Joe Oldham (Centre College), Timour Paltashev, James Parkerson (University of Arkansas), Shaunak Pawagi (SUNY at Stony Brook), Steve Pearce, Ted Pedersen

(University of Minnesota), Lu Peng (Louisiana State University), Gregory D. Peterson (The University of Tennessee), William Pierce (Hood College), Milos Prvulovic (Georgia Tech), Partha Ranganathan (HP Labs), Dejan Raskovic (University of Alaska, Fairbanks) Brad Richards (University of Puget Sound), Roman Rozanov, Louis Rubinfeld (Villanova University), Md Abdus Salam (Southern University), Augustine Samba (Kent State University), Robert Schaefer (Daniel Webster College), Carolyn J. C. Schauble (Colorado State University), Keith Schubert (CSU San Bernardino), William L. Schultz, Kelly Shaw (University of Richmond), Shahram Shirani (McMaster University), Scott Sigman (Drury University), Shai Simonson (Stonehill College), Bruce Smith, David Smith, Jeff W. Smith (University of Georgia, Athens), Mark Smotherman (Clemson University), Philip Snyder (Johns Hopkins University), Alex Sprintson (Texas A&M), Timothy D. Stanley (Brigham Young University), Dean Stevens (Morningside College), Nozar Tabrizi (Kettering University), Yuval Tamir (UCLA), Alexander Taubin (Boston University), Will Thacker (Winthrop University), Mithuna Thottethodi (Purdue University), Manghui Tu (Southern Utah University), Dean Tullsen (UC San Diego), Steve VanderLeest (Calvin College), Christopher Vickery (Queens College of CUNY), Rama Viswanathan (Beloit College), Ken Vollmar (Missouri State University), Guoping Wang (Indiana-Purdue University), Patricia Wenner (Bucknell University), Kent Wilken (University of California, Davis), David Wolfe (Gustavus Adolphus College), David Wood (University of Wisconsin, Madison), Ki Hwan Yum (University of Texas, San Antonio), Mohamed Zahran (City College of New York), Amr Zaky (Santa Clara University), Gerald D. Zarnett (Ryerson University), Nian Zhang (South Dakota School of Mines & Technology), Jiling Zhong (Troy University), Huiyang Zhou (North Carolina State University), Weiyu Zhu (Illinois Wesleyan University).

A special thanks also goes to **Mark Smotherman** for making multiple passes 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 **Katey Birtcher**, **Steve Merken**, and **Nate McFadden**: I certainly couldn't have completed the book without them. We also want to extend thanks to **Lisa Jones**, who managed the book production process, and **Victoria Pearson Esser**, who did the cover design. The cover cleverly connects the post-PC era content of this edition to the cover of the first edition.

Finally, I owe a huge debt to **Yunsup Lee** and **Andrew Waterman** for taking on this conversion to RISC-V in their spare time while founding a startup company. Kudos to **Eric Love** as well, who made RISC-V versions of the exercises in this edition while finishing his Ph.D. We're all excited to see what will happen with RISC-V in academia and beyond.

The contributions of the nearly 150 people we mentioned here have helped make this new edition what I hope will be our best book yet. Enjoy!

David A. Patterson

ACKNOWLEDGMENTS

RISC-V updates and contributions by

Andrew S. Waterman
SiFive, Inc.

Yunsup Lee
SiFive, Inc.

Additional contributions by

Perry Alexander
The University of Kansas

Peter J. Ashenden
Ashenden Designs Pty Ltd

Jason D. Bakos
University of South Carolina

Javier Diaz Bruguera
Universidade de Santiago de Compostela

Jichuan Chang
Google

Matthew Farrens
University of California, Davis

David Kaeli
Northeastern University

Nicole Kaiyan
University of Adelaide

David Kirk
NVIDIA

Zachary Kurmas
Grand Valley State University

James R. Larus
School of Computer and
Communications Science at EPFL

Jacob Leverich
Stanford University

Kevin Lim
Hewlett-Packard

Eric Love
University of California,
Berkeley

John Nickolls
NVIDIA

John Y. Oliver
Cal Poly, San Luis Obispo

Milos Prvulovic
Georgia Tech

Partha Ranganathan
Google

Mark Smotherman
Clemson University

Figures 1.7, 1.8 Courtesy of iFixit (www.ifixit.com).

Figure 1.9 Courtesy of Chipworks (www.chipworks.com).

Figure 1.13 Courtesy of Intel.

Figures 1.10.1, 1.10.2, 4.15.2 Courtesy of the Charles Babbage
Institute, University of Minnesota Libraries, Minneapolis.

Figures 1.10.3, 4.15.1, 4.15.3, 5.12.3, 6.14.2 Courtesy of IBM.

Figure 1.10.4 Courtesy of Cray Inc.

Figure 1.10.5 Courtesy of Apple Computer, Inc.

Figure 1.10.6 Courtesy of the Computer History Museum.

Figures 5.17.1, 5.17.2 Courtesy of Museum of Science, Boston.

Figure 5.17.4 Courtesy of MIPS Technologies, Inc.

Figure 6.15.1 Courtesy of NASA Ames Research Center.

David A. Patterson is the Pardee Professor of Computer Science, Emeritus at the University of California at Berkeley, which he joined after graduating from UCLA in 1977. His teaching has been honored by the Distinguished Teaching Award from the University of California, the Karlstrom Award from ACM, and the Mulligan Education Medal and Undergraduate Teaching Award from IEEE. Patterson received the IEEE Technical Achievement Award and the ACM Eckert-Mauchly Award for contributions to RISC, and he shared the IEEE Johnson Information Storage Award for contributions to RAID. He also shared the IEEE John von Neumann Medal and the C & C Prize with John Hennessy. Like his coauthor, Patterson is a Fellow of the American Academy of Arts and Sciences, the Computer History Museum, ACM, and IEEE, and he was elected to the National Academy of Engineering, the National Academy of Sciences, and the Silicon Valley Engineering Hall of Fame. He served on the Information Technology Advisory Committee to the US President, as chair of the CS division in the Berkeley EECS department, as chair of the Computing Research Association, and as President of ACM. This record led to Distinguished Service Awards from ACM, CRA, and SIGARCH.

At Berkeley, Patterson led the design and implementation of RISC I, likely the first VLSI reduced instruction set computer, and the foundation of the commercial SPARC architecture. He was a leader of the Redundant Arrays of Inexpensive Disks (RAID) project, which led to dependable storage systems from many companies. He was also involved in the Network of Workstations (NOW) project, which led to cluster technology used by Internet companies and later to cloud computing. These projects earned four dissertation awards from ACM. His current research projects are Algorithm-Machine-People and Algorithms and Specializers for Provably Optimal Implementations with Resilience and Efficiency. The AMP Lab is developing scalable machine learning algorithms, warehouse-scale-computer-friendly programming models, and crowd-sourcing tools to gain valuable insights quickly from big data in the cloud. The ASPIRE Lab uses deep hardware and software co-tuning to achieve the highest possible performance and energy efficiency for mobile and rack computing systems.

John L. Hennessy is a Professor of Electrical Engineering and Computer Science at Stanford University, where he has been a member of the faculty since 1977 and was, from 2000 to 2016, its tenth President. Hennessy is a Fellow of the IEEE and ACM; a member of the National Academy of Engineering, the National Academy of Science, and the American Philosophical Society; and a Fellow of the American Academy of Arts and Sciences. Among his many awards are the 2001 Eckert-Mauchly Award for his contributions to RISC technology, the 2001 Seymour Cray Computer Engineering Award, and the 2000 John von Neumann Award, which he shared with David Patterson. He has also received seven honorary doctorates.

In 1981, he started the MIPS project at Stanford with a handful of graduate students. After completing the project in 1984, he took a leave from the university to cofound MIPS Computer Systems (now MIPS Technologies), which developed one of the first commercial RISC microprocessors. As of 2006, over 2 billion MIPS microprocessors have been shipped in devices ranging from video games and palmtop computers to laser printers and network switches. Hennessy subsequently led the DASH (Director Architecture for Shared Memory) project, which prototyped the first scalable cache coherent multiprocessor; many of the key ideas have been adopted in modern multiprocessors. In addition to his technical activities and university responsibilities, he has continued to work with numerous start-ups, both as an early-stage advisor and an investor.