HZ BOOKS 华章教育
ELSEVIER

经 典 原 版 书 库

计算机体系结构

量化研究方法

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

（英文版·原书第6版）

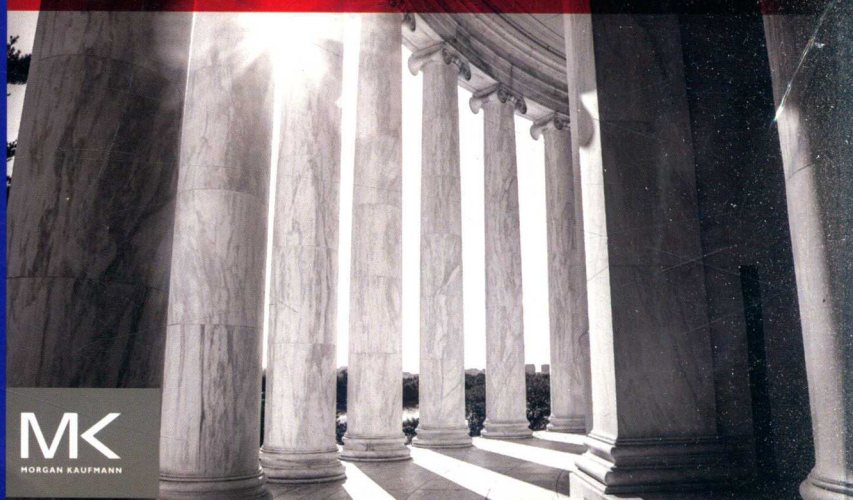
Computer Architecture
A Quantitative Approach, Sixth Edition

Sixth Edition

John L. Hennessy | David A. Patterson

COMPUTER ARCHITECTURE

A Quantitative Approach



机械工业出版社
China Machine Press

MK MORGAN KAUFMANN

经典原版书库

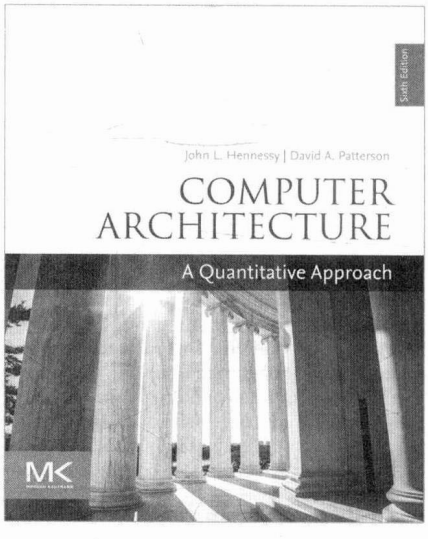
计算机体系结构

量化研究方法

（英文版·原书第6版）

Computer Architecture

A Quantitative Approach, Sixth Edition



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

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

计算机体系结构：量化研究方法（英文版·原书第6版）/（美）约翰·L. 亨尼斯（John L. Hennessy），（美）戴维·A. 帕特森（David A. Patterson）著. —北京：机械工业出版社，2019.7
（经典原版书库）
书名原文：Computer Architecture: A Quantitative Approach, Sixth Edition

ISBN 978-7-111-63110-1

I. 计… II. ①约… ②戴… III. 计算机体系结构－英文 IV. TP303

中国版本图书馆CIP数据核字（2019）第130605号

本书版权登记号：图字 01-2019-3869

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：北京文昌阁彩色印刷有限责任公司　　版　　次：2019年7月第1版第1次印刷
开　　本：186mm×240mm 1/16　　印　　张：58.25
书　　号：ISBN 978-7-111-63110-1　　定　　价：269.00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

Computer Architecture Formulas

1. *CPU time* = Instruction count × Clock cycles per instruction × Clock cycle time
2. X is n times faster than Y: n = Execution time$_Y$ / Execution time$_X$ = Performance$_X$ / Performance$_Y$
3. *Amdahl's Law:* $$\text{Speedup}_{\text{overall}} = \frac{\text{Execution time}_{\text{old}}}{\text{Execution time}_{\text{new}}} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$
4. $$\text{Energy}_{\text{dynamic}} \propto 1/2 \times \text{Capacitive load} \times \text{Voltage}^2$$
5. $$\text{Power}_{\text{dynamic}} \propto 1/2 \times \text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency switched}$$
6. $$\text{Power}_{\text{static}} \propto \text{Current}_{\text{static}} \times \text{Voltage}$$
7. *Availability* = Mean time to fail / (Mean time to fail + Mean time to repair)
8. *Die yield* = Wafer yield × 1 / (1 + Defects per unit area × Die area)^N

   where Wafer yield accounts for wafers that are so bad they need not be tested and N is a parameter called the process-complexity factor, a measure of manufacturing difficulty. N ranges from 11.5 to 15.5 in 2011.
9. *Means—arithmetic (AM), weighted arithmetic (WAM), and geometric (GM):*

$$\text{AM} = \frac{1}{n} \sum_{i=1}^{n} \text{Time}_i \quad \text{WAM} = \sum_{i=1}^{n} \text{Weight}_i \times \text{Time}_i \quad \text{GM} = \sqrt[n]{\prod_{i=1}^{n} \text{Time}_i}$$

   where Time$_i$ is the execution time for the ith program of a total of n in the workload, Weight$_i$ is the weighting of the ith program in the workload.
10. *Average memory-access time* = Hit time + Miss rate × Miss penalty
11. *Misses per instruction* = Miss rate × Memory access per instruction
12. *Cache index size:* 2^{index} = Cache size /(Block size × Set associativity)
13. *Power Utilization Effectiveness (PUE)* of a Warehouse Scale Computer = $\frac{\text{Total Facility Power}}{\text{IT Equipment Power}}$

Rules of Thumb

1. *Amdahl/Case Rule:* A balanced computer system needs about 1 MB of main memory capacity and 1 megabit per second of I/O bandwidth per MIPS of CPU performance.
2. *90/10 Locality Rule:* A program executes about 90% of its instructions in 10% of its code.
3. *Bandwidth Rule:* Bandwidth grows by at least the square of the improvement in latency.
4. *2:1 Cache Rule:* The miss rate of a direct-mapped cache of size N is about the same as a two-way set-associative cache of size $N/2$.
5. *Dependability Rule:* Design with no single point of failure.
6. *Watt-Year Rule*: The fully burdened cost of a Watt per year in a Warehouse Scale Computer in North America in 2011, including the cost of amortizing the power and cooling infrastructure, is about $2.

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，我们的联系方法如下：

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

华章教育

华章科技图书出版中心

In Praise of *Computer Architecture: A Quantitative Approach* Sixth Edition

"Although important concepts of architecture are timeless, this edition has been thoroughly updated with the latest technology developments, costs, examples, and references. Keeping pace with recent developments in open-sourced architecture, the instruction set architecture used in the book has been updated to use the RISC-V ISA."

—from the foreword by Norman P. Jouppi, Google

"*Computer Architecture: A Quantitative Approach* is a classic that, like fine wine, just keeps getting better. I bought my first copy as I finished up my undergraduate degree and it remains one of my most frequently referenced texts today."

—James Hamilton, Amazon Web Service

"Hennessy and Patterson wrote the first edition of this book when graduate students built computers with 50,000 transistors. Today, warehouse-size computers contain that many servers, each consisting of dozens of independent processors and billions of transistors. The evolution of computer architecture has been rapid and relentless, but *Computer Architecture: A Quantitative Approach* has kept pace, with each edition accurately explaining and analyzing the important emerging ideas that make this field so exciting."

—James Larus, Microsoft Research

"Another timely and relevant update to a classic, once again also serving as a window into the relentless and exciting evolution of computer architecture! The new discussions in this edition on the slowing of Moore's law and implications for future systems are must-reads for both computer architects and practitioners working on broader systems."

—Parthasarathy (Partha) Ranganathan, Google

"I love the 'Quantitative Approach' books because they are written by engineers, for engineers. John Hennessy and Dave Patterson show the limits imposed by mathematics and the possibilities enabled by materials science. Then they teach through real-world examples how architects analyze, measure, and compromise to build working systems. This sixth edition comes at a critical time: Moore's Law is fading just as deep learning demands unprecedented compute cycles. The new chapter on domain-specific architectures documents a number of promising approaches and prophesies a rebirth in computer architecture. Like the scholars of the European Renaissance, computer architects must understand our own history, and then combine the lessons of that history with new techniques to remake the world."

—Cliff Young, Google

Foreword

by Norman P. Jouppi, Google

Much of the improvement in computer performance over the last 40 years has been provided by computer architecture advancements that have leveraged Moore's Law and Dennard scaling to build larger and more parallel systems. Moore's Law is the observation that the maximum number of transistors in an integrated circuit doubles approximately every two years. Dennard scaling refers to the reduction of MOS supply voltage in concert with the scaling of feature sizes, so that as transistors get smaller, their power density stays roughly constant. With the end of Dennard scaling a decade ago, and the recent slowdown of Moore's Law due to a combination of physical limitations and economic factors, the sixth edition of the preeminent textbook for our field couldn't be more timely. Here are some reasons.

First, because domain-specific architectures can provide equivalent performance and power benefits of three or more historical generations of Moore's Law and Dennard scaling, they now can provide better implementations than may ever be possible with future scaling of general-purpose architectures. And with the diverse application space of computers today, there are many potential areas for architectural innovation with domain-specific architectures. Second, high-quality implementations of open-source architectures now have a much longer lifetime due to the slowdown in Moore's Law. This gives them more opportunities for continued optimization and refinement, and hence makes them more attractive. Third, with the slowing of Moore's Law, different technology components have been scaling heterogeneously. Furthermore, new technologies such as 2.5D stacking, new nonvolatile memories, and optical interconnects have been developed to provide more than Moore's Law can supply alone. To use these new technologies and nonhomogeneous scaling effectively, fundamental design decisions need to be reexamined from first principles. Hence it is important for students, professors, and practitioners in the industry to be skilled in a wide range of both old and new architectural techniques. All told, I believe this is the most exciting time in computer architecture since the industrial exploitation of instruction-level parallelism in microprocessors 25 years ago.

The largest change in this edition is the addition of a new chapter on domain-specific architectures. It's long been known that customized domain-specific architectures can have higher performance, lower power, and require less silicon area than general-purpose processor implementations. However when general-purpose

processors were increasing in single-threaded performance by 40% per year (see Fig. 1.11), the extra time to market required to develop a custom architecture vs. using a leading-edge standard microprocessor could cause the custom architecture to lose much of its advantage. In contrast, today single-core performance is improving very slowly, meaning that the benefits of custom architectures will not be made obsolete by general-purpose processors for a very long time, if ever. Chapter 7 covers several domain-specific architectures. Deep neural networks have very high computation requirements but lower data precision requirements – this combination can benefit significantly from custom architectures. Two example architectures and implementations for deep neural networks are presented: one optimized for inference and a second optimized for training. Image processing is another example domain; it also has high computation demands and benefits from lower-precision data types. Furthermore, since it is often found in mobile devices, the power savings from custom architectures are also very valuable. Finally, by nature of their reprogrammability, FPGA-based accelerators can be used to implement a variety of different domain-specific architectures on a single device. They also can benefit more irregular applications that are frequently updated, like accelerating internet search.

Although important concepts of architecture are timeless, this edition has been thoroughly updated with the latest technology developments, costs, examples, and references. Keeping pace with recent developments in open-sourced architecture, the instruction set architecture used in the book has been updated to use the RISC-V ISA.

On a personal note, after enjoying the privilege of working with John as a graduate student, I am now enjoying the privilege of working with Dave at Google. What an amazing duo!

Preface

Why We Wrote This Book

Through six editions of this book, our goal has been to describe the basic principles underlying what will be tomorrow's technological developments. Our excitement about the opportunities in computer architecture has not abated, and we echo what we said about the field in the first edition: "It is not a dreary science of paper machines that will never work. No! It's a discipline of keen intellectual interest, requiring the balance of marketplace forces to cost-performance-power, leading to glorious failures and some notable successes."

Our primary objective in writing our first book was to change the way people learn and think about computer architecture. We feel this goal is still valid and important. The field is changing daily and must be studied with real examples and measurements on real computers, rather than simply as a collection of definitions and designs that will never need to be realized. We offer an enthusiastic welcome to anyone who came along with us in the past, as well as to those who are joining us now. Either way, we can promise the same quantitative approach to, and analysis of, real systems.

As with earlier versions, we have strived to produce a new edition that will continue to be as relevant for professional engineers and architects as it is for those involved in advanced computer architecture and design courses. Like the first edition, this edition has a sharp focus on new platforms—personal mobile devices and warehouse-scale computers—and new architectures—specifically, domain-specific architectures. As much as its predecessors, this edition aims to demystify computer architecture through an emphasis on cost-performance-energy trade-offs and good engineering design. We believe that the field has continued to mature and move toward the rigorous quantitative foundation of long-established scientific and engineering disciplines.

This Edition

The ending of Moore's Law and Dennard scaling is having as profound effect on computer architecture as did the switch to multicore. We retain the focus on the extremes in size of computing, with personal mobile devices (PMDs) such as cell phones and tablets as the clients and warehouse-scale computers offering cloud computing as the server. We also maintain the other theme of parallelism in all its forms: *data-level parallelism (DLP)* in Chapters 1 and 4, *instruction-level parallelism (ILP)* in Chapter 3, *thread-level parallelism* in Chapter 5, and *request-level parallelism* (RLP) in Chapter 6.

The most pervasive change in this edition is switching from MIPS to the RISC-V instruction set. We suspect this modern, modular, open instruction set may become a significant force in the information technology industry. It may become as important in computer architecture as Linux is for operating systems.

The newcomer in this edition is Chapter 7, which introduces domain-specific architectures with several concrete examples from industry.

As before, the first three appendices in the book give basics on the RISC-V instruction set, memory hierarchy, and pipelining for readers who have not read a book like *Computer Organization and Design*. To keep costs down but still supply supplemental material that is of interest to some readers, available online at https://www.elsevier.com/books-and-journals/book-companion/9780128119051 are nine more appendices. There are more pages in these appendices than there are in this book!

This edition continues the tradition of using real-world examples to demonstrate the ideas, and the "Putting It All Together" sections are brand new. The "Putting It All Together" sections of this edition include the pipeline organizations and memory hierarchies of the ARM Cortex A8 processor, the Intel core i7 processor, the NVIDIA GTX-280 and GTX-480 GPUs, and one of the Google warehouse-scale computers.

Topic Selection and Organization

As before, we have taken a conservative approach to topic selection, for there are many more interesting ideas in the field than can reasonably be covered in a treatment of basic principles. We have steered away from a comprehensive survey of every architecture a reader might encounter. Instead, our presentation focuses on core concepts likely to be found in any new machine. The key criterion remains that of selecting ideas that have been examined and utilized successfully enough to permit their discussion in quantitative terms.

Our intent has always been to focus on material that is not available in equivalent form from other sources, so we continue to emphasize advanced content wherever possible. Indeed, there are several systems here whose descriptions cannot be found in the literature. (Readers interested strictly in a more basic introduction to computer architecture should read *Computer Organization and Design: The Hardware/Software Interface*.)

An Overview of the Content

Chapter 1 includes formulas for energy, static power, dynamic power, integrated circuit costs, reliability, and availability. (These formulas are also found on the front inside cover.) Our hope is that these topics can be used through the rest of the book. In addition to the classic quantitative principles of computer design and performance measurement, it shows the slowing of performance improvement of general-purpose microprocessors, which is one inspiration for domain-specific architectures.

Our view is that the instruction set architecture is playing less of a role today than in 1990, so we moved this material to Appendix A. It now uses the RISC-V architecture. (For quick review, a summary of the RISC-V ISA can be found on the back inside cover.) For fans of ISAs, Appendix K was revised for this edition and covers 8 RISC architectures (5 for desktop and server use and 3 for embedded use), the 80×86, the DEC VAX, and the IBM 360/370.

We then move onto memory hierarchy in Chapter 2, since it is easy to apply the cost-performance-energy principles to this material, and memory is a critical resource for the rest of the chapters. As in the past edition, Appendix B contains an introductory review of cache principles, which is available in case you need it. Chapter 2 discusses 10 advanced optimizations of caches. The chapter includes virtual machines, which offer advantages in protection, software management, and hardware management, and play an important role in cloud computing. In addition to covering SRAM and DRAM technologies, the chapter includes new material both on Flash memory and on the use of stacked die packaging for extending the memory hierarchy. The PIAT examples are the ARM Cortex A8, which is used in PMDs, and the Intel Core i7, which is used in servers.

Chapter 3 covers the exploitation of instruction-level parallelism in high-performance processors, including superscalar execution, branch prediction (including the new tagged hybrid predictors), speculation, dynamic scheduling, and simultaneous multithreading. As mentioned earlier, Appendix C is a review of pipelining in case you need it. Chapter 3 also surveys the limits of ILP. Like Chapter 2, the PIAT examples are again the ARM Cortex A8 and the Intel Core i7. While the third edition contained a great deal on Itanium and VLIW, this material is now in Appendix H, indicating our view that this architecture did not live up to the earlier claims.

The increasing importance of multimedia applications such as games and video processing has also increased the importance of architectures that can exploit data-level parallelism. In particular, there is a rising interest in computing using graphical processing units (GPUs), yet few architects understand how GPUs really work. We decided to write a new chapter in large part to unveil this new style of computer architecture. Chapter 4 starts with an introduction to vector architectures, which acts as a foundation on which to build explanations of multimedia SIMD instruction set extensions and GPUs. (Appendix G goes into even more depth on vector architectures.) This chapter introduces the Roofline performance model and then uses it to compare the Intel Core i7 and the NVIDIA GTX 280 and GTX 480 GPUs. The chapter also describes the Tegra 2 GPU for PMDs.

Chapter 5 describes multicore processors. It explores symmetric and distributed-memory architectures, examining both organizational principles and performance. The primary additions to this chapter include more comparison of multicore organizations, including the organization of multicore-multilevel caches, multicore coherence schemes, and on-chip multicore interconnect. Topics in synchronization and memory consistency models are next. The example is the Intel Core i7. Readers interested in more depth on interconnection networks should read Appendix F, and those interested in larger scale multiprocessors and scientific applications should read Appendix I.

Chapter 6 describes warehouse-scale computers (WSCs). It was extensively revised based on help from engineers at Google and Amazon Web Services. This chapter integrates details on design, cost, and performance of WSCs that few architects are aware of. It starts with the popular MapReduce programming model before describing the architecture and physical implementation of WSCs, including cost. The costs allow us to explain the emergence of cloud computing, whereby it can be cheaper to compute using WSCs in the cloud than in your local datacenter. The PIAT example is a description of a Google WSC that includes information published for the first time in this book.

The new Chapter 7 motivates the need for Domain-Specific Architectures (DSAs). It draws guiding principles for DSAs based on the four examples of DSAs. Each DSA corresponds to chips that have been deployed in commercial settings. We also explain why we expect a renaissance in computer architecture via DSAs given that single-thread performance of general-purpose microprocessors has stalled.

This brings us to Appendices A through M. Appendix A covers principles of ISAs, including RISC-V, and Appendix K describes 64-bit versions of RISC V, ARM, MIPS, Power, and SPARC and their multimedia extensions. It also includes some classic architectures (80x86, VAX, and IBM 360/370) and popular embedded instruction sets (Thumb-2, microMIPS, and RISC V C). Appendix H is related, in that it covers architectures and compilers for VLIW ISAs.

As mentioned earlier, Appendix B and Appendix C are tutorials on basic caching and pipelining concepts. Readers relatively new to caching should read Appendix B before Chapter 2, and those new to pipelining should read Appendix C before Chapter 3.

Appendix D, "Storage Systems," has an expanded discussion of reliability and availability, a tutorial on RAID with a description of RAID 6 schemes, and rarely found failure statistics of real systems. It continues to provide an introduction to queuing theory and I/O performance benchmarks. We evaluate the cost, performance, and reliability of a real cluster: the Internet Archive. The "Putting It All Together" example is the NetApp FAS6000 filer.

Appendix E, by Thomas M. Conte, consolidates the embedded material in one place.

Appendix F, on interconnection networks, is revised by Timothy M. Pinkston and José Duato. Appendix G, written originally by Krste Asanović, includes a description of vector processors. We think these two appendices are some of the best material we know of on each topic.

Appendix H describes VLIW and EPIC, the architecture of Itanium.

Appendix I describes parallel processing applications and coherence protocols for larger-scale, shared-memory multiprocessing. Appendix J, by David Goldberg, describes computer arithmetic.

Appendix L, by Abhishek Bhattacharjee, is new and discusses advanced techniques for memory management, focusing on support for virtual machines and design of address translation for very large address spaces. With the growth in clouds processors, these architectural enhancements are becoming more important.

Appendix M collects the "Historical Perspective and References" from each chapter into a single appendix. It attempts to give proper credit for the ideas in each chapter and a sense of the history surrounding the inventions. We like to think of this as presenting the human drama of computer design. It also supplies references that the student of architecture may want to pursue. If you have time, we recommend reading some of the classic papers in the field that are mentioned in these sections. It is both enjoyable and educational to hear the ideas directly from the creators. "Historical Perspective" was one of the most popular sections of prior editions.

Navigating the Text

There is no single best order in which to approach these chapters and appendices, except that all readers should start with Chapter 1. If you don't want to read everything, here are some suggested sequences:

- *Memory Hierarchy:* Appendix B, Chapter 2, and Appendices D and M.
- *Instruction-Level Parallelism:* Appendix C, Chapter 3, and Appendix H
- *Data-Level Parallelism:* Chapters 4, 6, and 7, Appendix G
- *Thread-Level Parallelism:* Chapter 5, Appendices F and I
- *Request-Level Parallelism:* Chapter 6
- *ISA:* Appendices A and K

Appendix E can be read at any time, but it might work best if read after the ISA and cache sequences. Appendix J can be read whenever arithmetic moves you. You should read the corresponding portion of Appendix M after you complete each chapter.

Chapter Structure

The material we have selected has been stretched upon a consistent framework that is followed in each chapter. We start by explaining the ideas of a chapter. These ideas are followed by a "Crosscutting Issues" section, a feature that shows how the ideas covered in one chapter interact with those given in other chapters. This is

followed by a "Putting It All Together" section that ties these ideas together by showing how they are used in a real machine.

Next in the sequence is "Fallacies and Pitfalls," which lets readers learn from the mistakes of others. We show examples of common misunderstandings and architectural traps that are difficult to avoid even when you know they are lying in wait for you. The "Fallacies and Pitfalls" sections is one of the most popular sections of the book. Each chapter ends with a "Concluding Remarks" section.

Case Studies With Exercises

Each chapter ends with case studies and accompanying exercises. Authored by experts in industry and academia, the case studies explore key chapter concepts and verify understanding through increasingly challenging exercises. Instructors should find the case studies sufficiently detailed and robust to allow them to create their own additional exercises.

Brackets for each exercise (<chapter.section>) indicate the text sections of primary relevance to completing the exercise. We hope this helps readers to avoid exercises for which they haven't read the corresponding section, in addition to providing the source for review. Exercises are rated, to give the reader a sense of the amount of time required to complete an exercise:

[10] Less than 5 min (to read and understand)

[15] 5–15 min for a full answer

[20] 15–20 min for a full answer

[25] 1 h for a full written answer

[30] Short programming project: less than 1 full day of programming

[40] Significant programming project: 2 weeks of elapsed time

[Discussion] Topic for discussion with others

Solutions to the case studies and exercises are available for instructors who register at *textbooks.elsevier.com*.

Supplemental Materials

A variety of resources are available online at https://www.elsevier.com/books/computer-architecture/hennessy/978-0-12-811905-1, including the following:

- Reference appendices, some guest authored by subject experts, covering a range of advanced topics
- Historical perspectives material that explores the development of the key ideas presented in each of the chapters in the text

- Instructor slides in PowerPoint
- Figures from the book in PDF, EPS, and PPT formats
- Links to related material on the Web
- List of errata

New materials and links to other resources available on the Web will be added on a regular basis.

Helping Improve This Book

Finally, it is possible to make money while reading this book. (Talk about cost performance!) If you read the Acknowledgments that follow, 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 (ca6bugs@mkp.com).

We welcome general comments to the text and invite you to send them to a separate email address at ca6comments@mkp.com.

Concluding Remarks

Once again, this book is a true co-authorship, with each of us writing half the chapters and an equal share of the appendices. We can't imagine how long it would have taken without someone else doing half the work, offering inspiration when the task seemed hopeless, providing the key insight to explain a difficult concept, supplying over-the-weekend reviews of chapters, and commiserating when the weight of our other obligations made it hard to pick up the pen.

Thus, once again, we share equally the blame for what you are about to read.

John Hennessy ■ *David Patterson*

Acknowledgments

Although this is only the sixth edition of this book, we have actually created ten different versions of the text: three versions of the first edition (alpha, beta, and final) and two versions of the second, third, and fourth editions (beta and final). Along the way, we have received help from hundreds of reviewers and users. Each of these people has helped make this book better. Thus, we have chosen to list all of the people who have made contributions to some version of this book.

Contributors to the Sixth Edition

Like prior editions, this is a community effort that involves scores of volunteers. Without their help, this edition would not be nearly as polished.

Reviewers

Jason D. Bakos, University of South Carolina; Rajeev Balasubramonian, University of Utah; Jose Delgado-Frias, Washington State University; Diana Franklin, The University of Chicago; Norman P. Jouppi, Google; Hugh C. Lauer, Worcester Polytechnic Institute; Gregory Peterson, University of Tennessee; Bill Pierce, Hood College; Parthasarathy Ranganathan, Google; William H. Robinson, Vanderbilt University; Pat Stakem, Johns Hopkins University; Cliff Young, Google; Amr Zaky, University of Santa Clara; Gerald Zarnett, Ryerson University; Huiyang Zhou, North Carolina State University.

Members of the University of California-Berkeley Par Lab and RAD Lab who gave frequent reviews of Chapters 1, 4, and 6 and shaped the explanation of GPUs and WSCs: Krste Asanović, Michael Armbrust, Scott Beamer, Sarah Bird, Bryan Catanzaro, Jike Chong, Henry Cook, Derrick Coetzee, Randy Katz, Yunsup Lee, Leo Meyervich, Mark Murphy, Zhangxi Tan, Vasily Volkov, and Andrew Waterman.

Appendices

Krste Asanović, University of California, Berkeley (Appendix G); Abhishek Bhattacharjee, Rutgers University (Appendix L); Thomas M. Conte, North Carolina State University (Appendix E); José Duato, Universitat Politècnica de

Valèncía and Simula (Appendix F); David Goldberg, Xerox PARC (Appendix J); Timothy M. Pinkston, University of Southern California (Appendix F).

José Flich of the Universidad Politécnica de Valencia provided significant contributions to the updating of Appendix F.

Case Studies With Exercises

Jason D. Bakos, University of South Carolina (Chapters 3 and 4); Rajeev Balasubramonian, University of Utah (Chapter 2); Diana Franklin, The University of Chicago (Chapter 1 and Appendix C); Norman P. Jouppi, Google, (Chapter 2); Naveen Muralimanohar, HP Labs (Chapter 2); Gregory Peterson, University of Tennessèe (Appendix A); Parthasarathy Ranganathan, Google (Chapter 6); Cliff Young, Google (Chapter 7); Amr Zaky, University of Santa Clara (Chapter 5 and Appendix B).

Jichuan Chang, Junwhan Ahn, Rama Govindaraju, and Milad Hashemi assisted in the development and testing of the case studies and exercises for Chapter 6.

Additional Material

John Nickolls, Steve Keckler, and Michael Toksvig of NVIDIA (Chapter 4 NVIDIA GPUs); Victor Lee, Intel (Chapter 4 comparison of Core i7 and GPU); John Shalf, LBNL (Chapter 4 recent vector architectures); Sam Williams, LBNL (Roofline model for computers in Chapter 4); Steve Blackburn of Australian National University and Kathryn McKinley of University of Texas at Austin (Intel performance and power measurements in Chapter 5); Luiz Barroso, Urs Hölzle, Jimmy Clidaris, Bob Felderman, and Chris Johnson of Google (the Google WSC in Chapter 6); James Hamilton of Amazon Web Services (power distribution and cost model in Chapter 6).

Jason D. Bakos of the University of South Carolina updated the lecture slides for this edition.

This book could not have been published without a publisher, of course. We wish to thank all the Morgan Kaufmann/Elsevier staff for their efforts and support. For this fifth edition, we particularly want to thank our editors Nate McFadden and Steve Merken, who coordinated surveys, development of the case studies and exercises, manuscript reviews, and the updating of the appendices.

We must also thank our university staff, Margaret Rowland and Roxana Infante, for countless express mailings, as well as for holding down the fort at Stanford and Berkeley while we worked on the book.

Our final thanks go to our wives for their suffering through increasingly early mornings of reading, thinking, and writing.