Preface

Why We Wrote This Book

Through five 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—multicore and GPUs. 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

We said the fourth edition of *Computer Architecture: A Quantitative Approach* may have been the most significant since the first edition due to the switch to multicore chips. The feedback we received this time was that the book had lost the sharp focus of the first edition, covering everthing equally but without emphasis and context. We're pretty sure that won't be said about the fifth edition.

We believe most of the excitement is at 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. (Observant readers may seen the hint for cloud computing on the cover.) We are struck by the common theme of these two extremes in cost, performance, and energy efficiency despite their difference in size. As a result, the running context through each chapter is computing for PMDs and for warehouse scale computers, and Chapter 6 is a brand-new chapter on the latter topic.

The other theme is parallelism in all its forms. We first idetify the two types of application-level parallelism in Chapter 1: *data-level parallelism (DLP)*, which arises because there are many data items that can be operated on at the same time, and *task-level parallelism (TLP)*, which arises because tasks of work are created that can operate independently and largely in parallel. We then explain the four architectural styles that exploit DLP and TLP: *instruction-level parallelism (ILP)* in Chapter 3; *vector architectures* and *graphic processor units (GPUs)* in Chapter 4, which is a brand-new chapter for this edition; *thread-level parallelism* in Chapter 5; and *request-level parallelism* (RLP) via warehouse-scale computers in Chapter 6, which is also a brand-new chapter for this edition. We moved memory hierarchy earlier in the book to Chapter 2, and we moved the storage systems chapter to Appendix D. We are particularly proud about Chapter 4, which contains the most detailed and clearest explanation of GPUs yet, and Chapter 6, which is the first publication of the most recent details of a Google Warehouse-scale computer.

As before, the first three appendices in the book give basics on the MIPS instruction set, memory hierachy, 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 are of interest to some readers, available online at http://booksite.mkp.com/9780123838728/ 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 has been beefed up in this edition. It 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, the PIAT section has been upgraded to use the new SPECPower benchmark.

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 still uses the MIPS64 architecture. (For quick review, a summary of the MIPS ISA can be found on the back inside cover.) For fans of ISAs, Appendix K covers 10 RISC architectures, the 80x86, 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 offers 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 on Flash memory. 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, speculation, dynamic scheduling, and 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 achitectures 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 instrution set extensions and GPUs. (Appendix G goes into even more depth on vector architectures.) The section on GPUs was the most difficult to write in this book, in that it took many iterations to get an accurate description that was also easy to understand. A significant challenge was the terminology. We decided to go with our own terms and then provide a translation between our terms and the official NVIDIA terms. (A copy of that table can be found in the back inside cover pages.) 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. Topics in synchronization and memory consistency models are next. The example is the Intel Core i7. Readers interested in interconnection networks on a chip should read Appendix F, and those interested in larger scale multiprocessors and scientific applications should read Appendix I.

As mentioned earlier, Chapter 6 describes the newest topic in computer architecture, warehouse-scale computers (WSCs). Based on help from engineers at Amazon Web Services and Google, 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 implemention 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.

This brings us to Appendices A through L. Appendix A covers principles of ISAs, including MIPS64, and Appendix K describes 64-bit versions of Alpha, MIPS, PowerPC, and SPARC and their multimedia extensions. It also includes some classic architectures (80x86, VAX, and IBM 360/370) and popular embedded instruction sets (ARM, Thumb, SuperH, MIPS16, and Mitsubishi M32R). Appendix H is related, in that it covers architectures and compilers for VLIW ISAs.

As mentioned earlier, Appendices B and 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, has been 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 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 Appendix D.
- *Instruction-Level Parallelism:* Appendix C, Chapter 3, and Appendix H
- *Data-Level Parallelism:* Chapters 4 and 6, 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 L 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 minutes (to read and understand)

[15] 5–15 minutes for a full answer

[20] 15–20 minutes for a full answer

[25] 1 hour 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 http://booksite.mkp.com/9780123838728/, 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 (*ca5bugs@mkp.com*).

We welcome general comments to the text and invite you to send them to a separate email address at *ca5comments@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 reviews over the weekend of chapters, and commiserating when the weight of our other obligations made it hard to pick up the pen. (These obligations have escalated exponentially with the number of editions, as the biographies attest.) Thus, once again we share equally the blame for what you are about to read.

John Hennessy ■ *David Patterson*