计 算 机 科 学 丛 书

Computer Organization & Design

The Hardware/Software Interface

计算机组织与设计
硬件/软件接口

(英文版·第2版)

(美) John L. Hennessy
David A. Patterson 著

机械工业出版社 China Machine Press
MORGAN KAUFMANN

计算机科学丛书

计算机组织与设计
硬件/软件接口

（英文版 第2版）

Computer Organization & Design
The Hardware/Software Interface

(Second Edition)

(美) John L. Hennessy
David A. Patterson 著

机械工业出版社
China Machine Press

本书版权登记号：图字：01-1999-2023

图书在版编目(CIP)数据

计算机组织与设计 硬件/软件接口：第2版：英文/(美)亨尼丝(Hennessy, J. L.), (美)帕蒂森(Patterson, D. A.)著.-北京：机械工业出版社，1999.9

(计算机科学丛书)

ISBN 7-111-07437-8

Ⅰ.计… Ⅱ.①亨… ②帕… Ⅲ.①电子计算机－系统结构－英文②电子计算机－系统设计－英文 Ⅳ.TP302

中国版本图书馆CIP数据核字(1999)第33792号

出 版 者：马九荣（北京市西城区百万庄大街22号 邮政编码 100037）

北京第二外国语学院印刷厂印刷·新华书店北京发行所发行

1999年9月第1版第1次印刷

787mm×1092mm1/16·62.25印张

印数：0 001-4 000册

定价：80.00元

About the Authors

David A. Patterson has been teaching computer architecture at the University of California, Berkeley, since joining the faculty in 1977, and he holds the Pardee Chair of Computer Science. Past chair of the CS division in the EECS department at Berkeley and the ACM SIG in computer architecture, he is currently chair of the Computing Research Association. His teaching has been honored by the ACM, the IEEE, and the University of California. Patterson has also received the IEEE Technical Achievement Award and the Outstanding Alumnus Award of the UCLA Computer Science Department. He is a member of the National Academy of Engineering and is a fellow of both the ACM the IEEE.

At Berkeley, Patterson led the design and implementation of RISC I, likely the first VLSI Reduced Instruction Set Computer. This research became the foundation of the SPARC, architecture, currently used by Fujitsu, Sun Microsystems, and Xerox. He was also a leader of the Redundant Arrays of Inexpensive Disks (RAID) project, which led to highperformance storage systems from many companies. These projects earned three distinguished dissertation awards from the ACM. His current research interests are in building novel microprocessors using Intelligent DRAM (IRAM).

John L. Hennessy teaches computer architecture at Stanford University, where he has been a member of the faculty since 1997. He is currently Dean of the School of Engineering and the Frederick Emmons Terman Professor of Engineering. Hennessy is a fellow of the IEEE and ACM, a member of the National Academy of Engineering, and a fellow of the American Academy of Arts and Sciences. He received the 1994 IEEE Piore Award for his contributions to the development of RISC technology.

Hennessy's original research group at Stanford developed many of the techniques now in commercial use for optimizing compilers. In 1981, he started the MIPS project at Stanford with a handful of graduate students. After completing the Project in 1984, he took a one-year leave from the university to co-found MIPS Computer Systems, which developed one of the first commercial RISC microprocessors. MIPS Computer Systems has since merged with Silicon Graphics, where Hennessy consults as Chief Architect. His recent research at Stanford focuses on the area of designing and exploiting multiprocessors. Most recently, he has been involved in the development of the DASH multiprocessor architecture, one of the first distributed shared-memory multiprocessors.

Foreword

by John H. Crawford
Intel Fellow, Director of Microprocessor Architecture
Intel Corporation, Santa Clara, California

Computer design is an exciting and competitive discipline. The microprocessor industry is on a treadmill where we double microprocessor performance every 18 months and double microprocessor complexity—measured by the number of transistors per chip—every 24 months. This unprecedented rate of change has been evident for the entire 25-year history of the microprocessor, and it promises to continue for many years to come as the creativity and energy of many people are harnessed to drive innovation ahead in spite of the challenge of ever-smaller dimensions. This book trains the student with the concepts needed to lay a solid foundation for joining this exciting field. More importantly, this book provides a framework for thinking about computer organization and design that will enable the reader to continue the lifetime of learning necessary for staying at the forefront of this competitive discipline.

The text focuses on the boundary between hardware and software and explores the levels of hardware in the vicinity of this boundary. This boundary is captured in a computer's architecture specification. It is a critical boundary for a successful computer product: an architect must define an interface that can be efficiently implemented by hardware and efficiently targeted by compilers. The interface must be able to retain these efficiencies for many generations of hardware and compiler technology, much of which will be unknown at the time the architecture is specified. This boundary is central to the discipline of computer design: it is where compilation (in software) ends and interpretation (in hardware) begins.

This book builds on introductory programming skills to introduce the concepts of assembly language programming and the tools needed for this task: the assembler, linker, and loader. Once these prerequisites are completed, the remainder of the book explores the first few levels of hardware below the architectural interface. The basic concepts are motivated and introduced with clear and intuitive examples, then elaborated into the "real stuff" used in today's modern microprocessors. For example, doing the laundry is used as an analogy in Chapter 6 to explain the basic concepts of pipelining, a key technique used in all modern computers. In Chapter 4, algorithms for the basic

floating-point arithmetic operators such as addition, multiplication, and division are first explained in decimal, then in binary, and finally they are elaborated into the best-known methods used for high-speed arithmetic in today's computers.

New to this edition are sections in each chapter entitled "Real Stuff." These sections describe how the concepts from the chapter are implemented in commercially successful products. These provide relevant, tangible examples of the concepts and reinforce their importance. As an example, the Real Stuff in Chapter 6, Enhancing Performance with Pipelining, provides an overview of a dynamically scheduled pipeline as implemented in both the IBM/Motorola PowerPC 604 and Intel's Pentium Pro microprocessor.

The history of computing is woven as a thread throughout the book to reward the reader with a glimpse of key successes from the brief history of this young discipline. The other side of history is reported in the Fallacies and Pitfalls section of each chapter. Since we can learn more from failure than from success, these sections provide a wealth of learning!

The authors are two of the most admired teachers, researchers, and practitioners of the art of computer design today. John Hennessy has straddled both sides of the hardware/software boundary, providing technical leadership for the legendary MIPS compiler as well as the MIPS hardware products through many generations. David Patterson was one of the original RISC proponents: he coined the acronym RISC, evangelized the case for RISC, and served as a key consultant on Sun Microsystem's SPARC line of processors. Continuing his talent for marketable acronyms, his next breakthrough was RAID (Redundant Arrays of Inexpensive Disks), which revolutionized the disk storage industry for large data servers, and then NOW (Networks of Workstations).

Like other great "software" products, this second edition went through an extensive beta testing program: 13 beta sites tested the draft manuscript in classes to "debug" the text. Changes from this testing have been incorporated into the "production" version.

Patterson and Hennessy have succeeded in taking the first edition of their excellent introductory textbook on computer design and making it even better. This edition retains all of the good points of the original, yet adds significant new content and some minor enhancements. What results is an outstanding introduction to the exciting field of computer design.

Worked Examples

Chapter 5: The Processor: Datapath and Control

Chapter 6: Enhancing Performance with Pipelining

Chapter 7: Large and Fast: Exploiting Memory Hierarchy

Chapter 8: Interfacing Processors and Peripherals

Chapter 9: Multiprocessors

Appendix A: Assemblers, Linkers, and the SPIM Simulator

Appendix B: The Basics of Logic Design

Appendix C: Mapping Control to Hardware

Computer Organization and Design Online

ENHANCED

All of the following resources are available at *http://www.mkp.com/cod2e.htm*.

Web Extensions

These materials are extensions of the book's content.

Web Extension I: Survey of RISC Architectures

Supplies current detailed information for several RISC architectures.

- Desktop RISC Architectures (Alpha, PA-RISC, MIPS, PowerPC, SPARC)
- Embedded RISC Architectures (ARM, Hitachi SH4, MIT M32R, MIPS 16, Thumb)

Web Extension II: Introducing C to Pascal Programmers

Provides Pascal programmers with a quick reference for understanding the C code in the text.

- Variable Declarations
- Assignment Statements
- Relational Expressions and Conditional Statements
- Loops
- Examples to Put It All Together
- Exercises

Web Extension III: Another Approach to Instruction Set Architecture—VAX

Presents an example of a CISC computer architecture for comparison with the MIPS architecture described in the text.

- VAX Operands and Addressing Modes
- Encoding VAX Instructions
- VAX Operations
- An Example to Put it All Together: swap

- A Longer Example: sort
- Fallacies and Pitfalls
- Historical Perspective and Further Reading
- Exercises

Supplements

This electronic support package includes files that can be viewed and downloaded in a number of formats.

Lecture slides

Electronic versions of text figures

Instructors Manual

Links to course home pages from selected schools

Instructions for using new DOS and Windows versions of PCspim simulators

Links to SPIM simulators (see page xviii)

Resources

Multiprocessors Page

Extends Chapter 9's coverage of real machines by providing links to companies that manufacture current multiprocessor machines.

Discussion Group

Provides readers with the opportunity to exchange ideas and information related to the book.

The SPIM Simulator

Developed by James R. Larus, the SPIM S20 is a software simulator that runs assembly language programs for the MIPS R2000/R3000 RISC computers. It can read and run MIPS a.out files (when compiled and running on a system containing a MIPS processor). SPIM is a self-contained system that contains a debugger and an interface to the operating system.

SPIM is portable; it has run on a DECStation 31000/51000, Sun 3, Sun 4, PC/RT, IBM RS/6000, HP Bobcat, HP Snake, and Sequent. Students can generate code for a simple, clean, orthogonal computer, regardless of the machine used. SPIM comes with complete source code and documentation of all instructions.

SPIM can be downloaded in versions for DOS, Windows, and UNIX, either from *www.mkp.com/cod2e.htm* or by direct ftp.

Retrieval of SPIM by ftp

SPIM is available for anonymous ftp from *ftp.cs.wisc.edu* in the file *pub/spim/spim.tar.Z* (this is a compressed tar file).

For those who are unfamiliar with command-line anonymous ftp, here are the steps to follow to get a copy of your preferred version of SPIM.

1. ftp to *ftp.cs.wisc.edu* from your computer:

```
% ftp ftp.cs.wisc.edu
```

2. The ftp server will respond and ask you to log in. Log in as anonymous and use your email address as a password:

```
Name (ftp.cs.wisc.edu:larus): anonymous
331 Guest login ok, send login or email address as password
Password:
```

3. The server will then print a welcome message. Change to the directory containing spim:

```
ftp> cd pub/spim
```

4. Set binary mode for the transfer (since the file is compressed):

```
ftp> binary
```

5. Choose the file appropriate for your machine and copy:

```
ftp> get spim.tar.Z (UNIX)
ftp> get PCspim.zip (Windows)
ftp> get PCspim-dos.zip (DOS)
```

6. Exit the ftp program:

```
ftp> quit
```

7. Uncompress and untar the file:

```
% uncompress spim.tar.Z
% tar xvf spim.tar
```

If the uncompression fails, you probably forgot to set binary (step 4). Try again. There are directions in the file README.

Preface

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

Albert Einstein, *What I Believe*, 1930

About This Book

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

Modern computer technology requires professionals of every computing specialty to understand both hardware and software. The interaction between hardware and software at a variety of levels also offers a framework for understanding the fundamentals of computing. Whether your primary interest is 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.

Traditionally, the competing influences of assembly language, organization, and design have encouraged books that consider each area as a distinct subset. In our view, such distinctions have increasingly lost meaning as computer technology has advanced. To truly understand the breadth of our field, it is important to understand the interdependencies among these topics.

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.

Changes for the Second Edition

We had six major goals for the second edition: tie the ideas from the book more closely to the real world; enhance how well the book works for beginners; extend the book material using the World Wide Web; improve quality; improve pedagogy; and finally, update the technical content to reflect changes

in the industry since the publication of the first edition in 1994—the conventional reason for a new edition.

First, to make the examples in the book even more concrete and connected with the real world, in each chapter we explained how the ideas were realized in the latest microprocessors from Intel or from IBM/Motorola. Hence you can learn how the mechanisms discussed are used in the computer on your desktop. Each chapter has a new section called "Real Stuff" that ties the ideas you read about to the machine you probably use everyday.

Second, we wanted the book to work better for readers interested in an overview of computer organization. Each chapter now has a list of the key terms discussed in the chapter, and we added a glossary of more than 300 definitions. We also rely on analogies from everyday life to explain subtleties of computers:

- commercial airplanes to show how performance differs if measured as bandwidth or latency
- the stealth of spies to explain procedure invocation and nesting
- plumbing to show how carry-lookahead logic works
- the laundry room to explain pipeline execution and hazards
- a desk in a library to demonstrate principles of memory hierarchy
- the management overhead as committees grow to illustrate the difficulty of achieving high performance in large-scale multiprocessors

More specifically, we added more assembly language programming examples and more explanation in each example to help the beginner understand assembly language programming in Chapters 3 and 4. We also added an introductory section to the pipelining chapter (Chapter 6) that allows understanding of the important ideas and issues in pipeline design without having to delve into the details of a pipelined datapath and control.

Our third goal was to go beyond the limitations of a printed book by adding descriptions and links on the World Wide Web. Throughout this book, you will often see the "Web Enhanced" icon shown at the left. Wherever this icon appears, you can go to *http://www.mkp.com/cod2e.htm* to find materials related to the text.

The WWW lets us give examples of recent, relevant machines so that you can see the latest versions of the ideas in the book. For example, we've added a new online appendix (Web Extension I) comparing RISC architectures. Other examples include links for specific references in the book to other sites; instructions on how to use PCspim, the new DOS and Windows versions of the SPIM simulator, as well as links to all the versions of SPIM; access to all the figures from the book; lecture slides; links to instructors' home pages; and an online Instructors Manual. We also included some appendices from the first edition (Web Extensions II and III) that you may find valuable. We intend to update these pages periodically to make new and better links.

Fourth, we wanted to significantly reduce the flaws that creep into a book during the revision process. The first edition of the book used beta testing to see which ideas worked well and which did not, and we were very happy with the improvements as a result. We did the same with the second edition. To further reduce the chances of bugs in the book, we gave ourselves a longer development cycle and involved many more computer architects in its preparation. First, Tod Amon completely revised all exercises, in part based on suggestions of exercises by a dozen instructors. The book now has 30% new exercises and another 30% that have been reworked for a total of 400. We believe that they are much more clearly worded than before and that there is sufficient variety for a broader group of students. Second, Kent Wilken carefully read the beta edition, suggesting hundreds of improvements. After we revised the beta edition, George Adams gave another very careful read of our revision, again making hundreds of useful suggestions. Finally, we reviewed the copyedit and read the page proof to try to catch mistakes that can creep in during the book production process. Although we are sure there must still be bugs for which you can get rewards, we believe this edition is far cleaner than the first.

The fifth goal was to improve the exposition of the ideas in the book, based on difficulties mentioned by readers of the first edition. We expanded the section of Chapter 3 explaining procedures, showing the procedure infrastructure in a longer sequence of examples. Chapter 4 has a longer description of carry lookahead and carry save adders. We simplified the explanation of the multicycle datapath in Chapter 5 by adding several registers. Chapter 6 actually got a good deal shorter by adding an overview section, since it allowed us to reduce the number of examples in the detailed pipelining sections. We also made numerous changes in the pipeline diagrams to make them easier to understand and more consistent. Chapter 7 was reorganized to put all caches together before moving to virtual memory and then translation buffers, coming back to the commonalities at the end. We also changed the emphasis from virtual memory as simply another level of the hierarchy to the hardware enforcer of protection. Chapter 8 was refocused to be more quantitative and design oriented. Chapter 9 was completely rewritten and retitled, reflecting the dramatic change in the parallel processing industry since 1994.

Finally, in the interval since the first edition of this book, a computer has run a program at the rate of 1 teraFLOPS—a trillion floating-point operations per second or a million floating-point operations per *microsecond*, another computer has played better chess than the best human being, and the whole world is more closely connected thanks to the World Wide Web. These events occurred in part because computer designers have first improved performance of a single computer by a factor of 100 in the last 10 years and then harnessed together many of them to achieve even greater performance. We have included descriptions of new ideas that helped make these miracles occur, such as

branch prediction and out-of-order execution in Chapter 6, multilevel and nonblocking caches in Chapter 7, switched networks and new buses in Chapter 8, and nonuniform-memory-access, shared-memory multiprocessors and clusters in Chapter 9.

Supplements and Web Extensions

ENHANCED

A directory of the Web supplements, extensions, and resources appears on page xvi. In it you'll find a complete electronic supplements package, as well as a variety of materials and resources designed to support this text, that you can access on the publisher's World Wide Web site at *www.mkp.com/ cod2e.htm*. Included in the supplements package is an online Instructors Manual. The Instructors Manual contents are available from the Web site with the exception of the solutions. Instructors should contact the publisher directly to obtain access to solutions.

If they prefer, instructors may choose a printed Instructors Manual that includes chapter objectives, teaching hints, and critical points for each chapter as well as solutions to the exercises. Instructors should contact the publisher directly to obtain the printed Instructors Manual.

Relationship to CA:AQA

Some readers may be familiar with *Computer Architecture: A Quantitative Approach*. Our motivation in writing that book was to describe the principles of computer architecture using solid engineering fundamentals and quantitative cost/performance trade-offs. We used an approach that combined examples and measurements, based on commercial systems, to create realistic design experiences. Our goal was to demonstrate that computer architecture could be learned using scientific methodologies instead of a descriptive approach.

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

Thus, we knew that this book had to be much more than a subset of the material in *Computer Architecture*. We've approached every topic in a new way. Topics shared between the books were written anew for this effort, while many other topics are presented here for the first time. To further ensure the uniqueness of *Computer Organization and Design*, we exchanged the writing responsibilities we assigned to ourselves for *Computer Architecture*. The topics

that Hennessy covered in the first book were written by Patterson in this one, and vice versa. Several of our reviewers suggested that we call this book "Computer Organization: A Conceptual Approach" to emphasize the significant differences from our other book. It is our hope that the reader will find new insights in every section, as well as a more tractable introduction to the abstractions and principles at work in a modern computer.

We were so happy with *Computer Organization and Design* that the second edition of *Computer Architecture* was revised to remove most of the introductory material, hence there is much less overlap today than with the first editions of both books.

Learning by Evolution

It is tempting for authors to present the latest version of a hardware concept and spend considerable time explaining how these often sophisticated ideas work. We decided instead to present each idea from its first principles, emphasizing the simplest version of an idea, how it works, and how it came to be. We believe that presenting the fundamental concepts first offers greater insight into why machines look the way they do today, as well as how they might evolve as technology changes.

To facilitate this approach, we have based the book upon the MIPS processor. It offers an easy-to-understand instruction set and can be implemented in a simple, straightforward manner. This allows readers to grasp an entire machine organization and to follow exactly how the machine implements its instructions. Throughout the text, we present the concepts before the details, building from simpler versions of ideas to more complex ones. Examples of this approach can be found in almost every chapter. Chapter 3 builds up to MIPS assembly language starting with one simple instruction type. The concepts and algorithms used in modern computer arithmetic are built up starting from the familiar grade school algorithms in Chapter 4. Chapters 5 and 6 start from the simplest possible implementation of a MIPS subset and build to a fully pipelined version. Chapter 7 illustrates the abstractions and concepts in memory hierarchies by starting with the simplest possible cache, then extending it, and then covering virtual memory and TLBs using the same ideas.

This evolutionary process is used extensively in Chapters 5 and 6, where the complete datapath and control for a processor are presented. Since learning is a visual process, we have included sequences of figures that contain progressively more detail or show a sequence of events within the machine. We have also used a second color to help readers follow the figures and sequences of figures.

Learning from this Book

Our objective of demonstrating first principles through the interrelationship of hardware and software is enhanced by several features found in each