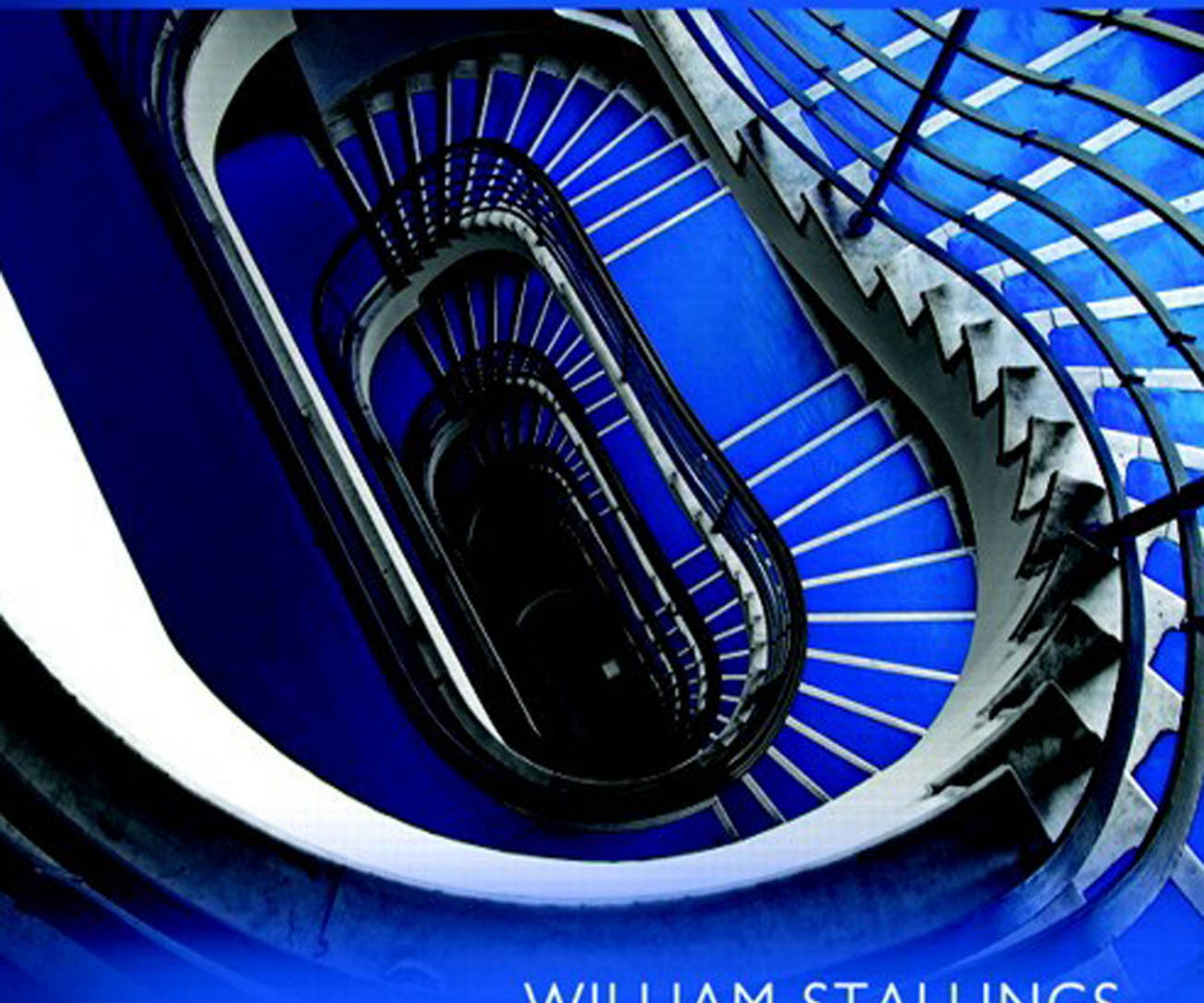
COMPUTER ORGANIZATION AND ARCHITECTURE

Designing for Performance

Tenth Edition



WILLIAM STALLINGS

Computer Organization and Architecture

Designing for Performance

Tenth Edition

This page intentionally left blank

Computer Organization and Architecture
Designing for Performance
Tenth Edition

William Stallings

With contribution by
Peter Zeno
University of Bridgeport

With Foreword by
Chris Jesshope
Professor (emeritus) University of Amsterdam

PEARSON

Boston • Columbus • Hoboken • Indianapolis • New York • San Francisco
Amsterdam • Cape Town • Dubai • London • Madrid • Milan • Munich • Paris • Montreal
Toronto • Delhi • Mexico City • São Paulo • Sydney • Hong Kong • Seoul • Singapore • Taipei • Tokyo

Vice President and Editorial Director, ECS: *Marcia J. Horton*
Executive Editor: *Tracy Johnson (Dunkelberger)*
Editorial Assistant: *Kelsey Loanes*
Program Manager: *Carole Snyder*
Director of Product Management: *Erin Gregg*
Team Lead Product Management: *Scott Disanno*
Project Manager: *Robert Engelhardt*
Media Team Lead: *Steve Wright*
R&P Manager: *Rachel Youdelman*
R&P Senior Project Manager: *Timothy Nicholls*
Procurement Manager: *Mary Fischer*
Senior Specialist, Program Planning and Support: *Maura Zaldivar-Garcia*
Inventory Manager: *Bruce Boundy*
VP of Marketing: *Christy Lesko*
Director of Field Marketing: *Demetrius Hall*
Product Marketing Manager: *Bram van Kempen*
Marketing Assistant: *Jon Bryant*
Cover Designer: *Marta Samsel*
Cover Art: © *anderm / Fotolia*
Full-Service Project Management: *Mahalatchoumy Saravanan, Jouve India*
Printer/Binder: *Edwards Brothers Malloy*
Cover Printer: *Lehigh-Phoenix Color/Hagerstown*
Typeface: *Times Ten LT Std 10/12*

Pearson Education Ltd., *London*
Pearson Education Australia Ply. Ltd., Sydney
Pearson Education Singapore, Pte. Ltd.
Pearson Education North Asia Ltd., *Hong Kong*
Pearson Education Canada, Inc., Toronto
Pearson Education de Mexico, S.A. de C.V.
Pearson Education–Japan, Tokyo
Pearson Education Malaysia, Pte. Ltd.
Pearson Education, Inc., *Hoboken, New Jersey*

Library of Congress Cataloging-in-Publication Data

Stallings, William.
Computer organization and architecture : designing for performance / William Stallings. — Tenth edition.
pages cm
Includes bibliographical references and index.
ISBN 978-0-13-410161-3 — ISBN 0-13-410161-8 1. Computer organization. 2. Computer architecture. I. Title.
QA76.9.C643S73 2016
004.2'2—dc23

2014044367

10 9 8 7 6 5 4 3 2 1

PEARSON

www.pearsonhighered.com

ISBN-10: 0-13-410161-8
ISBN-13: 978-0-13-410161-3

To Tricia
my loving wife, the kindest
and gentlest person

This page intentionally left blank

CONTENTS

[1]Online chapters, appendices, and other documents are Premium Content, available via the access card at the front of this book.

This page intentionally left blank

FOREWORD

by Chris Jesshope
Professor (emeritus) University of Amsterdam
Author of Parallel Computers (with R W Hockney), 1981 & 1988

Having been active in computer organization and architecture for many years, it is a pleasure to write this foreword for the new edition of William Stallings' comprehensive book on this subject. In doing this, I found myself reflecting on the trends and changes in this subject over the time that I have been involved in it. I myself became interested in computer architecture at a time of significant innovation and disruption. That disruption was brought about not only through advances in technology but perhaps more significantly through access to that technology. VLSI was here and VLSI design was available to students in the classroom. These were exciting times. The ability to integrate a mainframe style computer on a single silicon chip was a milestone, but that this was accomplished by an academic research team made the achievement quite unique. This period was characterized by innovation and diversity in computer architecture with one of the main trends being in the area of parallelism. In the 1970s, I had hands-on experience of the Illiac IV, which was an early example of explicit parallelism in computer architecture and which incidentally pioneered all semiconductor memory. This interaction, and it certainly was that, kick-started my own interest in computer architecture and organization, with particular emphasis on explicit parallelism in computer architecture.

Throughout the 1980s and early 1990s research flourished in this field and there was a great deal of innovation, much of which came to market through university start-ups. Ironically however, it was the same technology that reversed this trend. Diversity was gradually replaced with a near monoculture in computer systems with advances in just a few instruction set architectures. Moore's law, a self-fulfilling prediction that became an industry guideline, meant that basic device speeds and integration densities both grew exponentially, with the latter doubling every 18 months of so. The speed increase was the proverbial free lunch for computer architects and the integration levels allowed more complexity and innovation at the micro-architecture level. The free lunch of course did have a cost, that being the exponential growth of capital investment required to fulfill Moore's law, which once again limited the access to state-of-the-art technologies. Moreover, most users found it easier to wait for the next generation of mainstream processor than to invest in the innovations in parallel computers, with their pitfalls and difficulties. The exceptions to this were the few large institutions requiring ultimate performance; two topical examples being large-scale scientific simulation such as climate modeling and also in our security services for code breaking. For

everyone else, the name of the game was compatibility and two instruction set architectures that benefited from this were x86 and ARM, the latter in embedded systems and the former in just about everything else. Parallelism was still there in the implementation of these ISAs, it was just that it was implicit, harnessed by the architecture not in the instruction stream that drives it.

Throughout the late 1990s and early 2000s, this approach to implicitly exploiting concurrency in single-core computer systems flourished. However, in spite of the exponential growth of logic density, it was the cost of the techniques exploited which brought this era to a close. In superscalar processors, the logic costs do not grow linearly with issue width (parallelism), while some components grow as the square or even the cube of the issue width. Although the exponential growth in logic could sustain this continued development, there were two major pitfalls: it was increasingly difficult to expose concurrency implicitly from imperative programs and hence efficiencies in the use of instruction issue slots decreased. Perhaps more importantly, technology was experiencing a new barrier to performance gains, namely that of power dissipation, and several superscalar developments were halted because the silicon in them would have been too hot. These constraints have mandated the exploitation of explicit parallelism, despite the compatibility challenges. So it seems that again innovation and diversity are opening up this area to new research.

Perhaps not since the 1980s has it been so interesting to study in this field. That diversity is an economic reality can be seen by the decrease in issue width (implicit parallelism) and increase in the number of cores (explicit parallelism) in mainstream processors. However, the question is how to exploit this, both at the application and the system level. There are significant challenges here still to be solved. Superscalar processors rely on the processor to extract parallelism from a single instruction stream. What if we shifted the emphasis and provided an instruction stream with maximum parallelism, how can we exploit this in different configurations and/or generations of processors that require different levels of explicit parallelism? Is it possible therefore to have a micro-architecture that sequentializes and schedules this maximum concurrency captured in the ISA to match the current configuration of cores so that we gain the same compatibility in a world of explicit parallelism? Does this require operating systems in silicon for efficiency?

These are just some of the questions facing us today. To answer these questions and more requires a sound foundation in computer organization and architecture, and this book by William Stallings provides a very timely and comprehensive foundation. It gives a complete introduction to the basics required, tackling what can be quite complex topics with apparent simplicity. Moreover, it deals with the more recent developments in this field, where innovation has in the past, and is, currently taking place. Examples are in superscalar issue and in explicitly parallel multicores. What is more, this latest edition includes two very recent topics in the design and use of GPUs for general-purpose use and the latest trends in cloud computing, both of which have become mainstream only recently. The book makes good use of examples throughout to highlight the theoretical issues covered, and most of these examples are drawn from developments in the two most widely used ISAs, namely the x86 and ARM. To reiterate, this book is complete and is a pleasure to read and hopefully will kick-start more young researchers down the same path that I have enjoyed over the last 40 years!

PREFACE

WHAT'S NEW IN THE TENTH EDITION

Since the ninth edition of this book was published, the field has seen continued innovations and improvements. In this new edition, I try to capture these changes while maintaining a broad and comprehensive coverage of the entire field. To begin this process of revision, the ninth edition of this book was extensively reviewed by a number of professors who teach the subject and by professionals working in the field. The result is that, in many places, the narrative has been clarified and tightened, and illustrations have been improved.

Beyond these refinements to improve pedagogy and user-friendliness, there have been substantive changes throughout the book. Roughly the same chapter organization has been retained, but much of the material has been revised and new material has been added. The most noteworthy changes are as follows:

- **GPGPU [General-Purpose Computing on Graphics Processing Units (GPUs)]:** One of the most important new developments in recent years has been the broad adoption of GPGPUs to work in coordination with traditional CPUs to handle a wide range of applications involving large arrays of data. A new chapter is devoted to the topic of GPGPUs.
- **Heterogeneous multicore processors:** The latest development in multicore architecture is the heterogeneous multicore processor. A new section in the chapter on multicore processors surveys the various types of heterogeneous multicore processors.
- **Embedded systems:** The overview of embedded systems in Chapter 1 has been substantially revised and expanded to reflect the current state of embedded technology.
- **Microcontrollers:** In terms of numbers, almost all computers now in use are embedded microcontrollers. The treatment of embedded systems in Chapter 1 now includes coverage of microcontrollers. The ARM Cortex-M3 microcontroller is used as an example system throughout the text.
- **Cloud computing:** New to this edition is a discussion of cloud computing, with an overview in Chapter 1 and more detailed treatment in Chapter 17.
- **System performance:** The coverage of system performance issues has been revised, expanded, and reorganized for a clearer and more thorough treatment. Chapter 2 is devoted to this topic, and the issue of system performance arises throughout the book.

- **Flash memory:** The coverage of flash memory has been updated and expanded, and now includes a discussion of the technology and organization of flash memory for internal memory (Chapter 5) and external memory (Chapter 6).
- **Nonvolatile RAM:** New to this edition is treatment of three important new nonvolatile solid-state RAM technologies that occupy different positions in the memory hierarchy: STT-RAM, PCRAM, and ReRAM.
- **Direct cache access (DCA):** To meet the protocol processing demands for very high speed network connections, Intel and other manufacturers have developed DCA technologies that provide much greater throughput than traditional direct memory access (DMA) approaches. New to this edition, Chapter 7 explores DCA in some detail.
- **Intel Core Microarchitecture:** As in the previous edition, the Intel x86 family is used as a major example system throughout. The treatment has been updated to reflect newer Intel systems, especially the Intel Core Microarchitecture, which is used on both PC and server products.
- **Homework problems:** The number of supplemental homework problems, with solutions, available for student practice has been expanded.

SUPPORT OF ACM/IEEE COMPUTER SCIENCE CURRICULA 2013

The book is intended for both an academic and a professional audience. As a textbook, it is intended as a one- or two-semester undergraduate course for computer science, computer engineering, and electrical engineering majors. This edition is designed to support the recommendations of the ACM/IEEE Computer Science Curricula 2013 (CS2013). CS2013 divides all course work into three categories: Core-Tier 1 (all topics should be included in the curriculum); Core-Tier-2 (all or almost all topics should be included); and Elective (desirable to provide breadth and depth). In the Architecture and Organization (AR) area, CS2013 includes five Tier-2 topics and three Elective topics, each of which has a number of subtopics. This text covers all eight topics listed by CS2013. Table P.1 shows the support for the AR Knowledge Area provided in this textbook.

Table P.1 Coverage of CS2013 Architecture and Organization (AR) Knowledge Area

IAS Knowledge Units	Topics	Textbook Coverage
Digital Logic and Digital Systems (Tier 2)	• Overview and history of computer architecture • Combinational vs. sequential logic/Field programmable gate arrays as a fundamental combinational sequential logic building block • Multiple representations/layers of interpretation (hardware is just another layer) • Physical constraints (gate delays, fan-in, fan-out, energy/power)	—Chapter 1 —Chapter 11
Machine Level Representation of Data (Tier 2)	• Bits, bytes, and words • Numeric data representation and number bases • Fixed- and floating-point systems • Signed and twos-complement representations • Representation of non-numeric data (character codes, graphical data)	—Chapter 9 —Chapter 10

IAS Knowledge Units	Topics	Textbook Coverage
Assembly Level Machine Organization (Tier 2)	• Basic organization of the von Neumann machine • Control unit; instruction fetch, decode, and execution • Instruction sets and types (data manipulation, control, I/O) • Assembly/machine language programming • Instruction formats • Addressing modes • Subroutine call and return mechanisms (cross-reference PL/Language Translation and Execution) • I/O and interrupts • Shared memory multiprocessors/multicore organization • Introduction to SIMD vs. MIMD and the Flynn Taxonomy	—Chapter 1 —Chapter 7 —Chapter 12 —Chapter 13 —Chapter 17 —Chapter 18 —Chapter 20 —Chapter 21 —Appendix A
Memory System Organization and Architecture (Tier 2)	• Storage systems and their technology • Memory hierarchy: temporal and spatial locality • Main memory organization and operations • Latency, cycle time, bandwidth, and interleaving • Cache memories (address mapping, block size, replacement and store policy) • Multiprocessor cache consistency/Using the memory system for inter-core synchronization/atomic memory operations • Virtual memory (page table, TLB) • Fault handling and reliability	—Chapter 4 —Chapter 5 —Chapter 6 —Chapter 8 —Chapter 17
Interfacing and Communication (Tier 2)	• I/O fundamentals: handshaking, buffering, programmed I/O, interrupt-driven I/O • Interrupt structures: vectored and prioritized, interrupt acknowledgment • External storage, physical organization, and drives • Buses: bus protocols, arbitration, direct-memory access (DMA) • RAID architectures	—Chapter 3 —Chapter 6 —Chapter 7
Functional Organization (Elective)	• Implementation of simple datapaths, including instruction pipelining, hazard detection, and resolution • Control unit: hardwired realization vs. microprogrammed realization • Instruction pipelining • Introduction to instruction-level parallelism (ILP)	—Chapter 14 —Chapter 16 —Chapter 20 —Chapter 21
Multiprocessing and Alternative Architectures (Elective)	• Example SIMD and MIMD instruction sets and architectures • Interconnection networks • Shared multiprocessor memory systems and memory consistency • Multiprocessor cache coherence	—Chapter 12 —Chapter 13 —Chapter 17
Performance Enhancements (Elective)	• Superscalar architecture • Branch prediction, Speculative execution, Out-of-order execution • Prefetching • Vector processors and GPUs • Hardware support for multithreading • Scalability	—Chapter 15 —Chapter 16 —Chapter 19

OBJECTIVES

This book is about the structure and function of computers. Its purpose is to present, as clearly and completely as possible, the nature and characteristics of modern-day computer systems.

This task is challenging for several reasons. First, there is a tremendous variety of products that can rightly claim the name of computer, from single-chip microprocessors costing a few dollars to supercomputers costing tens of millions of dollars. Variety is exhibited not only in cost but also in size, performance, and application. Second, the rapid pace of change that has always characterized computer technology continues with no letup. These changes cover all aspects of computer technology, from the underlying integrated circuit technology used to construct computer components to the increasing use of parallel organization concepts in combining those components.

In spite of the variety and pace of change in the computer field, certain fundamental concepts apply consistently throughout. The application of these concepts depends on the current state of the technology and the price/performance objectives of the designer. The intent of this book is to provide a thorough discussion of the fundamentals of computer organization and architecture and to relate these to contemporary design issues.

The subtitle suggests the theme and the approach taken in this book. It has always been important to design computer systems to achieve high performance, but never has this requirement been stronger or more difficult to satisfy than today. All of the basic performance characteristics of computer systems, including processor speed, memory speed, memory capacity, and interconnection data rates, are increasing rapidly. Moreover, they are increasing at different rates. This makes it difficult to design a balanced system that maximizes the performance and utilization of all elements. Thus, computer design increasingly becomes a game of changing the structure or function in one area to compensate for a performance mismatch in another area. We will see this game played out in numerous design decisions throughout the book.

A computer system, like any system, consists of an interrelated set of components. The system is best characterized in terms of structure—the way in which components are interconnected, and function—the operation of the individual components. Furthermore, a computer's organization is hierarchical. Each major component can be further described by decomposing it into its major subcomponents and describing their structure and function. For clarity and ease of understanding, this hierarchical organization is described in this book from the top down:

- **Computer system:** Major components are processor, memory, I/O.
- **Processor:** Major components are control unit, registers, ALU, and instruction execution unit.
- **Control unit:** Provides control signals for the operation and coordination of all processor components. Traditionally, a microprogramming implementation has been used, in which major components are control memory, microinstruction sequencing logic, and registers. More recently, microprogramming has been less prominent but remains an important implementation technique.

The objective is to present the material in a fashion that keeps new material in a clear context. This should minimize the chance that the reader will get lost and should provide better motivation than a bottom-up approach.

Throughout the discussion, aspects of the system are viewed from the points of view of both architecture (those attributes of a system visible to a machine language programmer) and organization (the operational units and their interconnections that realize the architecture).

EXAMPLE SYSTEMS

This text is intended to acquaint the reader with the design principles and implementation issues of contemporary operating systems. Accordingly, a purely conceptual or theoretical treatment would be inadequate. To illustrate the concepts and to tie them to real-world design choices that must be made, two processor families have been chosen as running examples:

- **Intel x86 architecture:** The x86 architecture is the most widely used for nonembedded computer systems. The x86 is essentially a complex instruction set computer (CISC) with some RISC features. Recent members of the x86 family make use of superscalar and multicore design principles. The evolution of features in the x86 architecture provides a unique case-study of the evolution of most of the design principles in computer architecture.
- **ARM:** The ARM architecture is arguably the most widely used embedded processor, used in cell phones, iPods, remote sensor equipment, and many other devices. The ARM is essentially a reduced instruction set computer (RISC). Recent members of the ARM family make use of superscalar and multicore design principles.

Many, but by no means all, of the examples in this book are drawn from these two computer families. Numerous other systems, both contemporary and historical, provide examples of important computer architecture design features.

PLAN OF THE TEXT

The book is organized into six parts:

- Overview
- The computer system
- Arithmetic and logic
- The central processing unit
- Parallel organization, including multicore
- The control unit

The book includes a number of pedagogic features, including the use of interactive simulations and numerous figures and tables to clarify the discussion. Each chapter includes a list of key words, review questions, homework problems, and suggestions for further reading. The book also includes an extensive glossary, a list of frequently used acronyms, and a bibliography.

INSTRUCTOR SUPPORT MATERIALS

Support materials for instructors are available at the **Instructor Resource Center (IRC)** for this textbook, which can be reached through the publisher's Web site www.pearsonhighered.com/stallings or by clicking on the link labeled "Pearson Resources for Instructors" at this

book's Companion Web site at WilliamStallings.com/ComputerOrganization. To gain access to the IRC, please contact your local Pearson sales representative via pearsonhighered.com/educator/replocator/requestSalesRep.page or call Pearson Faculty Services at 1-800-526-0485. The IRC provides the following materials:

- **Projects manual:** Project resources including documents and portable software, plus suggested project assignments for all of the project categories listed subsequently in this Preface.
- **Solutions manual:** Solutions to end-of-chapter Review Questions and Problems.
- **PowerPoint slides:** A set of slides covering all chapters, suitable for use in lecturing.
- **PDF files:** Copies of all figures and tables from the book.
- **Test bank:** A chapter-by-chapter set of questions.
- **Sample syllabuses:** The text contains more material than can be conveniently covered in one semester. Accordingly, instructors are provided with several sample syllabuses that guide the use of the text within limited time. These samples are based on real-world experience by professors with the first edition.

The **Companion Web site**, at WilliamStallings.com/ComputerOrganization (click on Instructor Resources link) includes the following:

- Links to Web sites for other courses being taught using this book.
- Sign-up information for an Internet mailing list for instructors using this book to exchange information, suggestions, and questions with each other and with the author.

STUDENT RESOURCES

For this new edition, a tremendous amount of original supporting material for students has been made available online, at two Web locations. The **Companion Web Site**, at WilliamStallings.com/ComputerOrganization (click on Student Resources link), includes a list of relevant links organized by chapter and an errata sheet for the book.

Purchasing this textbook new grants the reader six months of access to the **Premium Content Site**, which includes the following materials:

- **Online chapters:** To limit the size and cost of the book, two chapters of the book are provided in PDF format. The chapters are listed in this book's table of contents.
- **Online appendices:** There are numerous interesting topics that support material found in the text but whose inclusion is not warranted in the printed text. A total of 13 appendices cover these topics for the interested student. The appendices are listed in this book's table of contents.
- **Homework problems and solutions:** To aid the student in understanding the material, a separate set of homework problems with solutions are available. Students can enhance their understanding of the material by working out the solutions to these problems and then checking their answers.

To access the Premium Content site, click on the *Premium Content* link at the Companion Web site or at pearsonhighered.com/stallings and enter the student access code found on the card in the front of the book.

Finally, I maintain the Computer Science Student Resource Site at **WilliamStallings.com/StudentSupport.html**.

PROJECTS AND OTHER STUDENT EXERCISES

For many instructors, an important component of a computer organization and architecture course is a project or set of projects by which the student gets hands-on experience to reinforce concepts from the text. This book provides an unparalleled degree of support for including a projects component in the course. The instructor's support materials available through Prentice Hall not only includes guidance on how to assign and structure the projects but also includes a set of user's manuals for various project types plus specific assignments, all written especially for this book. Instructors can assign work in the following areas:

- **Interactive simulation assignments:** Described subsequently.
- **Research projects:** A series of research assignments that instruct the student to research a particular topic on the Internet and write a report.
- **Simulation projects:** The IRC provides support for the use of the two simulation packages: SimpleScalar can be used to explore computer organization and architecture design issues. SMPCache provides a powerful educational tool for examining cache design issues for symmetric multiprocessors.
- **Assembly language projects:** A simplified assembly language, CodeBlue, is used and assignments based on the popular Core Wars concept are provided.
- **Reading/report assignments:** A list of papers in the literature, one or more for each chapter, that can be assigned for the student to read and then write a short report.
- **Writing assignments:** A list of writing assignments to facilitate learning the material.
- **Test bank:** Includes T/F, multiple choice, and fill-in-the-blank questions and answers.

This diverse set of projects and other student exercises enables the instructor to use the book as one component in a rich and varied learning experience and to tailor a course plan to meet the specific needs of the instructor and students. See Appendix A in this book for details.

INTERACTIVE SIMULATIONS

An important feature in this edition is the incorporation of interactive simulations. These simulations provide a powerful tool for understanding the complex design features of a modern computer system. A total of 20 interactive simulations are used to illustrate key functions and algorithms in computer organization and architecture design. At the relevant point in the book, an icon indicates that a relevant interactive simulation is available online for student use. Because the animations enable the user to set initial conditions, they can

serve as the basis for student assignments. The instructor's supplement includes a set of assignments, one for each of the animations. Each assignment includes several specific problems that can be assigned to students.

For access to the animations, click on the rotating globe at this book's Web site at http://williamstallings.com/ComputerOrganization.

ACKNOWLEDGMENTS

This new edition has benefited from review by a number of people, who gave generously of their time and expertise. The following professors and instructors reviewed all or a large part of the manuscript: Molisa Derk (Dickinson State University), Yaohang Li (Old Dominion University), Dwayne Ockel (Regis University), Nelson Luiz Passos (Midwestern State University), Mohammad Abdus Salam (Southern University), and Vladimir Zwass (Fairleigh Dickinson University).

Thanks also to the many people who provided detailed technical reviews of one or more chapters: Rekai Gonzalez Alberquilla, Allen Baum, Jalil Boukhobza, Dmitry Bufistov, Humberto Calderón, Jesus Carretero, Ashkan Eghbal, Peter Glaskowsky, Ram Huggahalli, Chris Jesshope, Athanasios Kakarountas, Isil Oz, Mitchell Poplingher, Roger Shepherd, Jigar Savla, Karl Stevens, Siri Uppalapati, Dr. Sriram Vajapeyam, Kugan Vivekanandarajah, Pooria M. Yaghini, and Peter Zeno,

Peter Zeno also contributed Chapter 19 on GPGPUs.

Professor Cindy Norris of Appalachian State University, Professor Bin Mu of the University of New Brunswick, and Professor Kenrick Mock of the University of Alaska kindly supplied homework problems.

Aswin Sreedhar of the University of Massachusetts developed the interactive simulation assignments and also wrote the test bank.

Professor Miguel Angel Vega Rodriguez, Professor Dr. Juan Manuel Sánchez Pérez, and Professor Dr. Juan Antonio Gómez Pulido, all of University of Extremadura, Spain, prepared the SMPCache problems in the instructor's manual and authored the SMPCache User's Guide.

Todd Bezenek of the University of Wisconsin and James Stine of Lehigh University prepared the SimpleScalar problems in the instructor's manual, and Todd also authored the SimpleScalar User's Guide.

Finally, I would like to thank the many people responsible for the publication of the book, all of whom did their usual excellent job. This includes the staff at Pearson, particularly my editor Tracy Johnson, her assistant Kelsey Loanes, program manager Carole Snyder, and production manager Bob Engelhardt. I also thank Mahalatchoumy Saravanan and the production staff at Jouve India for another excellent and rapid job. Thanks also to the marketing and sales staffs at Pearson, without whose efforts this book would not be in front of you.

About the Author

Dr. William Stallings has authored 17 textbooks, and counting revised editions, over 40 books on computer security, computer networking, and computer architecture. In over 30 years in the field, he has been a technical contributor, technical manager, and an executive with several high-technology firms. Currently, he is an independent consultant whose clients have included computer and networking manufacturers and customers, software development firms, and leading-edge government research institutions. He has 13 times received the award for the best computer science textbook of the year from the Text and Academic Authors Association.

He created and maintains the Computer Science Student Resource Site at ComputerScienceStudent.com. This site provides documents and links on a variety of subjects of general interest to computer science students (and professionals). His articles appear regularly at networking.answers.com, where he is the Networking Category Expert Writer. He is a member of the editorial board of *Cryptologia*, a scholarly journal devoted to all aspects of cryptology.

Dr. Stallings holds a PhD from MIT in computer science and a BS from Notre Dame in electrical engineering.

This page intentionally left blank

PART ONE INTRODUCTION

CHAPTER 1

BASIC CONCEPTS AND COMPUTER EVOLUTION

LEARNING OBJECTIVES

After studying this chapter, you should be able to:

- Explain the general functions and structure of a digital computer.
- Present an overview of the evolution of computer technology from early digital computers to the latest microprocessors.
- Present an overview of the evolution of the x86 architecture.
- Define embedded systems and list some of the requirements and constraints that various embedded systems must meet.

1.1 ORGANIZATION AND ARCHITECTURE

In describing computers, a distinction is often made between *computer architecture* and *computer organization*. Although it is difficult to give precise definitions for these terms, a consensus exists about the general areas covered by each. For example, see [VRAN80], [SIEW82], and [BELL78a]; an interesting alternative view is presented in [REDD76].

Computer architecture refers to those attributes of a system visible to a programmer or, put another way, those attributes that have a direct impact on the logical execution of a program. A term that is often used interchangeably with computer architecture is **instruction set architecture (ISA)**. The ISA defines instruction formats, instruction opcodes, registers, instruction and data memory; the effect of executed instructions on the registers and memory; and an algorithm for controlling instruction execution. **Computer organization** refers to the operational units and their interconnections that realize the architectural specifications. Examples of architectural attributes include the instruction set, the number of bits used to represent various data types (e.g., numbers, characters), I/O mechanisms, and techniques for addressing memory. Organizational attributes include those hardware details transparent to the programmer, such as control signals; interfaces between the computer and peripherals; and the memory technology used.

For example, it is an architectural design issue whether a computer will have a multiply instruction. It is an organizational issue whether that instruction will be implemented by a special multiply unit or by a mechanism that makes repeated use of the add unit of the system. The organizational decision may be based on the anticipated frequency of use of the multiply instruction, the relative speed of the two approaches, and the cost and physical size of a special multiply unit.

Historically, and still today, the distinction between architecture and organization has been an important one. Many computer manufacturers offer a family of computer models, all with the same architecture but with differences in organization. Consequently, the different models in the family have different price and performance characteristics. Furthermore, a particular architecture may span many years and encompass a number of different computer models, its organization changing with changing technology. A prominent example of both these phenomena is the IBM System/370 architecture. This architecture was first introduced in 1970 and

included a number of models. The customer with modest requirements could buy a cheaper, slower model and, if demand increased, later upgrade to a more expensive, faster model without having to abandon software that had already been developed. Over the years, IBM has introduced many new models with improved technology to replace older models, offering the customer greater speed, lower cost, or both. These newer models retained the same architecture so that the customer's software investment was protected. Remarkably, the System/370 architecture, with a few enhancements, has survived to this day as the architecture of IBM's mainframe product line.

In a class of computers called microcomputers, the relationship between architecture and organization is very close. Changes in technology not only influence organization but also result in the introduction of more powerful and more complex architectures. Generally, there is less of a requirement for generation-to-generation compatibility for these smaller machines. Thus, there is more interplay between organizational and architectural design decisions. An intriguing example of this is the reduced instruction set computer (RISC), which we examine in Chapter 15.

This book examines both computer organization and computer architecture. The emphasis is perhaps more on the side of organization. However, because a computer organization must be designed to implement a particular architectural specification, a thorough treatment of organization requires a detailed examination of architecture as well.

1.2 STRUCTURE AND FUNCTION

A computer is a complex system; contemporary computers contain millions of elementary electronic components. How, then, can one clearly describe them? The key is to recognize the hierarchical nature of most complex systems, including the computer [SIMO96]. A hierarchical system is a set of interrelated subsystems, each of the latter, in turn, hierarchical in structure until we reach some lowest level of elementary subsystem.

The hierarchical nature of complex systems is essential to both their design and their description. The designer need only deal with a particular level of the system at a time. At each level, the system consists of a set of components and their interrelationships. The behavior at each level depends only on a simplified, abstracted characterization of the system at the next lower level. At each level, the designer is concerned with structure and function:

- **Structure:** The way in which the components are interrelated.
- **Function:** The operation of each individual component as part of the structure.

In terms of description, we have two choices: starting at the bottom and building up to a complete description, or beginning with a top view and decomposing the system into its subparts. Evidence from a number of fields suggests that the top-down approach is the clearest and most effective [WEIN75].

The approach taken in this book follows from this viewpoint. The computer system will be described from the top down. We begin with the major components of a computer, describing their structure and function, and proceed to successively

lower layers of the hierarchy. The remainder of this section provides a very brief overview of this plan of attack.

Function

Both the structure and functioning of a computer are, in essence, simple. In general terms, there are only four basic functions that a computer can perform:

- **Data processing:** Data may take a wide variety of forms, and the range of processing requirements is broad. However, we shall see that there are only a few fundamental methods or types of data processing.
- **Data storage:** Even if the computer is processing data on the fly (i.e., data come in and get processed, and the results go out immediately), the computer must temporarily store at least those pieces of data that are being worked on at any given moment. Thus, there is at least a short-term data storage function. Equally important, the computer performs a long-term data storage function. Files of data are stored on the computer for subsequent retrieval and update.
- **Data movement:** The computer's operating environment consists of devices that serve as either sources or destinations of data. When data are received from or delivered to a device that is directly connected to the computer, the process is known as *input–output* (*I/O*), and the device is referred to as a *peripheral*. When data are moved over longer distances, to or from a remote device, the process is known as *data communications*.
- **Control:** Within the computer, a control unit manages the computer's resources and orchestrates the performance of its functional parts in response to instructions.

The preceding discussion may seem absurdly generalized. It is certainly possible, even at a top level of computer structure, to differentiate a variety of functions, but to quote [SIEW82]:

> There is remarkably little shaping of computer structure to fit the function to be performed. At the root of this lies the general-purpose nature of computers, in which all the functional specialization occurs at the time of programming and not at the time of design.

Structure

We now look in a general way at the internal structure of a computer. We begin with a traditional computer with a single processor that employs a microprogrammed control unit, then examine a typical multicore structure.

SIMPLE SINGLE-PROCESSOR COMPUTER Figure 1.1 provides a hierarchical view of the internal structure of a traditional single-processor computer. There are four main structural components:

- **Central processing unit (CPU):** Controls the operation of the computer and performs its data processing functions; often simply referred to as **processor**.
- **Main memory:** Stores data.

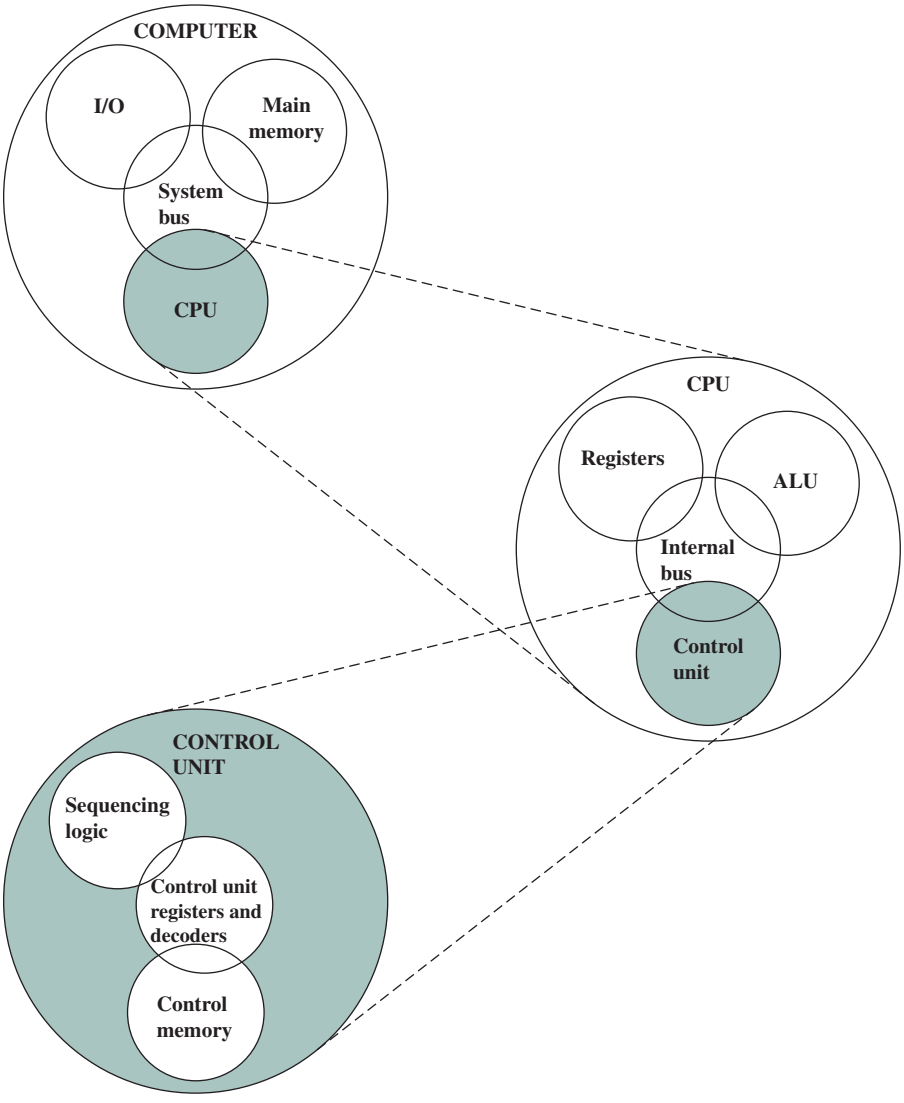


Figure 1.1 The Computer: Top-Level Structure

- **I/O:** Moves data between the computer and its external environment.
- **System interconnection:** Some mechanism that provides for communication among CPU, main memory, and I/O. A common example of system interconnection is by means of a **system bus**, consisting of a number of conducting wires to which all the other components attach.

There may be one or more of each of the aforementioned components. Traditionally, there has been just a single processor. In recent years, there has been increasing use of multiple processors in a single computer. Some design issues relating to multiple processors crop up and are discussed as the text proceeds; Part Five focuses on such computers.

Each of these components will be examined in some detail in Part Two. However, for our purposes, the most interesting and in some ways the most complex component is the CPU. Its major structural components are as follows:

- **Control unit:** Controls the operation of the CPU and hence the computer.
- **Arithmetic and logic unit (ALU):** Performs the computer's data processing functions.
- **Registers:** Provides storage internal to the CPU.
- **CPU interconnection:** Some mechanism that provides for communication among the control unit, ALU, and registers.

Part Three covers these components, where we will see that complexity is added by the use of parallel and pipelined organizational techniques. Finally, there are several approaches to the implementation of the control unit; one common approach is a *microprogrammed* implementation. In essence, a microprogrammed control unit operates by executing microinstructions that define the functionality of the control unit. With this approach, the structure of the control unit can be depicted, as in Figure 1.1. This structure is examined in Part Four.

MULTICORE COMPUTER STRUCTURE As was mentioned, contemporary computers generally have multiple processors. When these processors all reside on a single chip, the term *multicore computer* is used, and each processing unit (consisting of a control unit, ALU, registers, and perhaps cache) is called a *core*. To clarify the terminology, this text will use the following definitions.

- **Central processing unit (CPU):** That portion of a computer that fetches and executes instructions. It consists of an ALU, a control unit, and registers. In a system with a single processing unit, it is often simply referred to as a *processor*.
- **Core:** An individual processing unit on a processor chip. A core may be equivalent in functionality to a CPU on a single-CPU system. Other specialized processing units, such as one optimized for vector and matrix operations, are also referred to as cores.
- **Processor:** A physical piece of silicon containing one or more cores. The processor is the computer component that interprets and executes instructions. If a processor contains multiple cores, it is referred to as a **multicore processor**.

After about a decade of discussion, there is broad industry consensus on this usage.

Another prominent feature of contemporary computers is the use of multiple layers of memory, called *cache memory*, between the processor and main memory. Chapter 4 is devoted to the topic of cache memory. For our purposes in this section, we simply note that a cache memory is smaller and faster than main memory and is used to speed up memory access, by placing in the cache data from main memory, that is likely to be used in the near future. A greater performance improvement may be obtained by using multiple levels of cache, with level 1 (L1) closest to the core and additional levels (L2, L3, and so on) progressively farther from the core. In this scheme, level n is smaller and faster than level $n + 1$.

Figure 1.2 is a simplified view of the principal components of a typical multicore computer. Most computers, including embedded computers in smartphones and tablets, plus personal computers, laptops, and workstations, are housed on a motherboard. Before describing this arrangement, we need to define some terms. A **printed circuit board (PCB)** is a rigid, flat board that holds and interconnects chips and other electronic components. The board is made of layers, typically two to ten, that interconnect components via copper pathways that are etched into the board. The main printed circuit board in a computer is called a system board or **motherboard**, while smaller ones that plug into the slots in the main board are called expansion boards.

The most prominent elements on the motherboard are the chips. A **chip** is a single piece of semiconducting material, typically silicon, upon which electronic circuits and logic gates are fabricated. The resulting product is referred to as an **integrated circuit**.

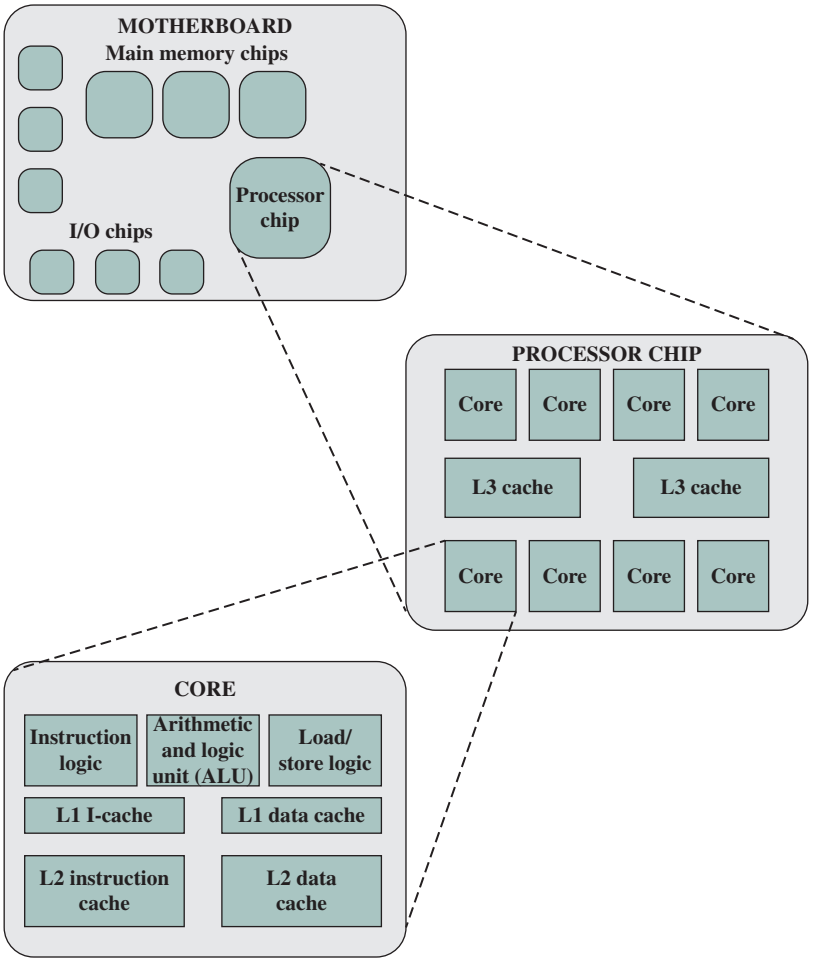


Figure 1.2 Simplified View of Major Elements of a Multicore Computer

The motherboard contains a slot or socket for the processor chip, which typically contains multiple individual cores, in what is known as a *multicore processor*. There are also slots for memory chips, I/O controller chips, and other key computer components. For desktop computers, expansion slots enable the inclusion of more components on expansion boards. Thus, a modern motherboard connects only a few individual chip components, with each chip containing from a few thousand up to hundreds of millions of transistors.

Figure 1.2 shows a processor chip that contains eight cores and an L3 cache. Not shown is the logic required to control operations between the cores and the cache and between the cores and the external circuitry on the motherboard. The figure indicates that the L3 cache occupies two distinct portions of the chip surface. However, typically, all cores have access to the entire L3 cache via the aforementioned control circuits. The processor chip shown in Figure 1.2 does not represent any specific product, but provides a general idea of how such chips are laid out.

Next, we zoom in on the structure of a single core, which occupies a portion of the processor chip. In general terms, the functional elements of a core are:

- **Instruction logic:** This includes the tasks involved in fetching instructions, and decoding each instruction to determine the instruction operation and the memory locations of any operands.
- **Arithmetic and logic unit (ALU):** Performs the operation specified by an instruction.
- **Load/store logic:** Manages the transfer of data to and from main memory via cache.

The core also contains an L1 cache, split between an instruction cache (I-cache) that is used for the transfer of instructions to and from main memory, and an L1 data cache, for the transfer of operands and results. Typically, today's processor chips also include an L2 cache as part of the core. In many cases, this cache is also split between instruction and data caches, although a combined, single L2 cache is also used.

Keep in mind that this representation of the layout of the core is only intended to give a general idea of internal core structure. In a given product, the functional elements may not be laid out as the three distinct elements shown in Figure 1.2, especially if some or all of these functions are implemented as part of a microprogrammed control unit.

EXAMPLES It will be instructive to look at some real-world examples that illustrate the hierarchical structure of computers. Figure 1.3 is a photograph of the motherboard for a computer built around two Intel Quad-Core Xeon processor chips. Many of the elements labeled on the photograph are discussed subsequently in this book. Here, we mention the most important, in addition to the processor sockets:

- PCI-Express slots for a high-end display adapter and for additional peripherals (Section 3.6 describes PCIe).
- Ethernet controller and Ethernet ports for network connections.
- USB sockets for peripheral devices.

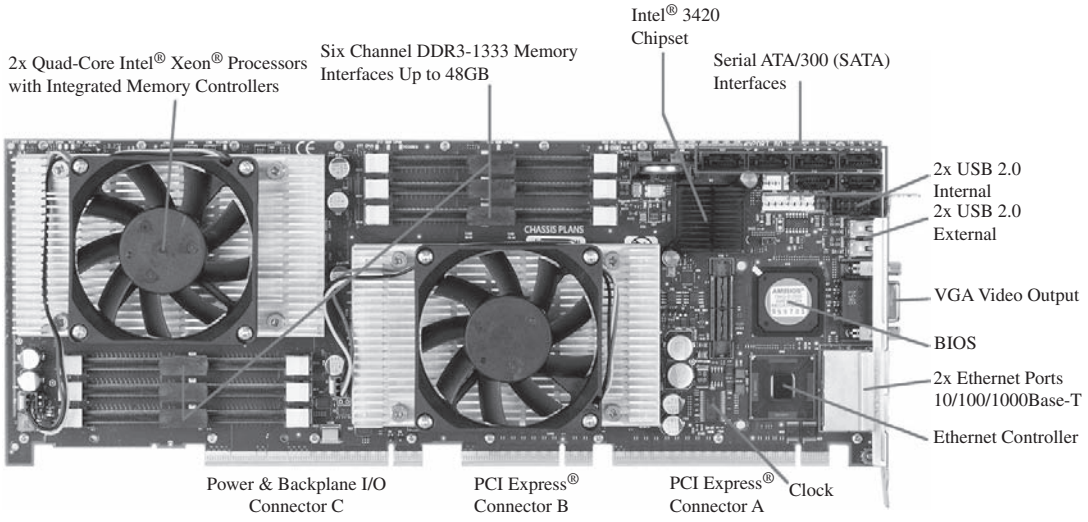


Figure 1.3 Motherboard with Two Intel Quad-Core Xeon Processors
Source: Chassis Plans, www.chassis-plans.com

- Serial ATA (SATA) sockets for connection to disk memory (Section 7.7 discusses Ethernet, USB, and SATA).
- Interfaces for DDR (double data rate) main memory chips (Section 5.3 discusses DDR).
- Intel 3420 chipset is an I/O controller for direct memory access operations between peripheral devices and main memory (Section 7.5 discusses DDR).

Following our top-down strategy, as illustrated in Figures 1.1 and 1.2, we can now zoom in and look at the internal structure of a processor chip. For variety, we look at an IBM chip instead of the Intel processor chip. Figure 1.4 is a photograph of the processor chip for the IBM zEnterprise EC12 mainframe computer. This chip has 2.75 billion transistors. The superimposed labels indicate how the silicon real estate of the chip is allocated. We see that this chip has six cores, or processors. In addition, there are two large areas labeled L3 cache, which are shared by all six processors. The L3 control logic controls traffic between the L3 cache and the cores and between the L3 cache and the external environment. Additionally, there is storage control (SC) logic between the cores and the L3 cache. The memory controller (MC) function controls access to memory external to the chip. The GX I/O bus controls the interface to the channel adapters accessing the I/O.

Going down one level deeper, we examine the internal structure of a single core, as shown in the photograph of Figure 1.5. Keep in mind that this is a portion of the silicon surface area making up a single-processor chip. The main sub-areas within this core area are the following:

- **ISU (instruction sequence unit):** Determines the sequence in which instructions are executed in what is referred to as a superscalar architecture (Chapter 16).
- **IFU (instruction fetch unit):** Logic for fetching instructions.