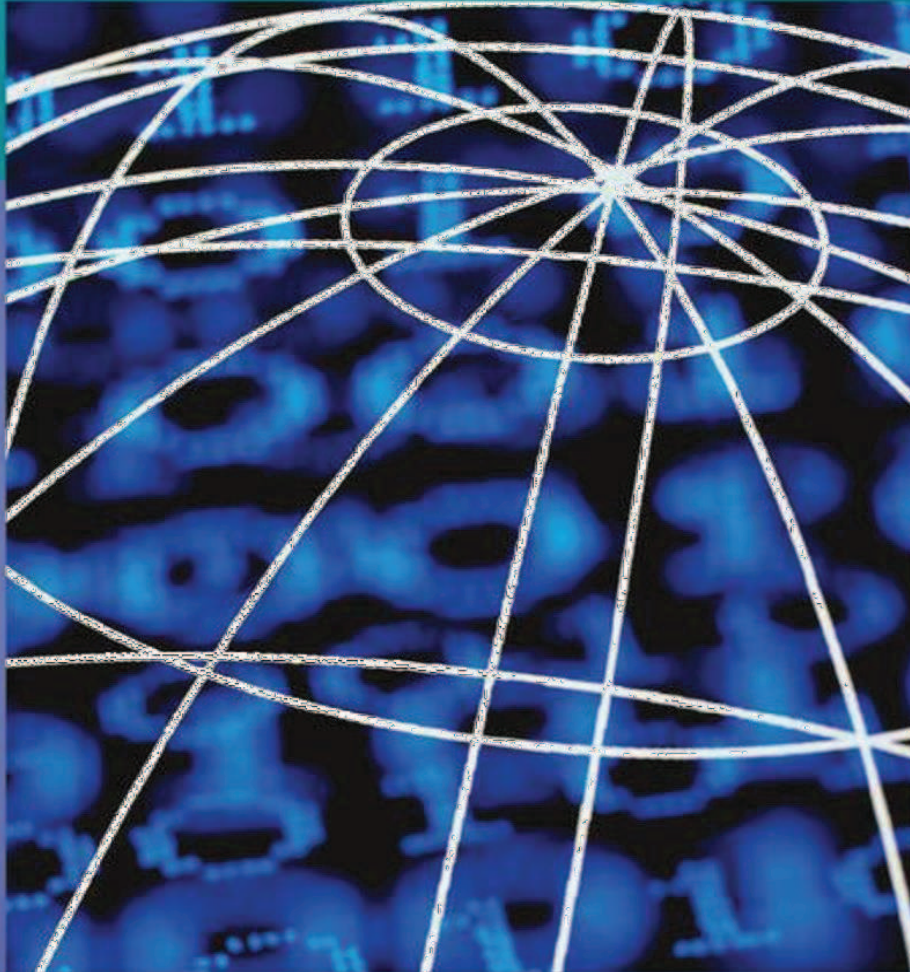


Carl Hamacher • Zvonko Vranesic • Safwat Zaky • Naraig Manjikian



COMPUTER ORGANIZATION AND EMBEDDED SYSTEMS

Sixth Edition

This page intentionally left blank

COMPUTER ORGANIZATION AND EMBEDDED SYSTEMS

This page intentionally left blank

COMPUTER ORGANIZATION AND EMBEDDED SYSTEMS

SIXTH EDITION

Carl Hamacher

Queen's University

Zvonko Vranesic

University of Toronto

Safwat Zaky

University of Toronto

Naraig Manjikian

Queen's University





COMPUTER ORGANIZATION AND EMBEDDED SYSTEMS, SIXTH EDITION

Published by McGraw-Hill, a business unit of The McGraw-Hill Companies, Inc., 1221 Avenue of the Americas, New York, NY 10020. Copyright © 2012 by The McGraw-Hill Companies, Inc. All rights reserved. Previous editions 2002, 1996, and 1990. No part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written consent of The McGraw-Hill Companies, Inc., including, but not limited to, in any network or other electronic storage or transmission, or broadcast for distance learning.

Some ancillaries, including electronic and print components, may not be available to customers outside the United States.

This book is printed on acid-free paper.

1 2 3 4 5 6 7 8 9 DOC/DOC 0 9 8 7 6 5 4 3 2 1

ISBN 978-0-07-338065-0

MHID 0-07-338065-2

Vice President & Editor-in-Chief: *Marty Lange*

Vice President EDP/Central Publishing Services: *Kimberly Meriwether David*

Publisher: *Raghothaman Srinivasan*

Senior Sponsoring Editor: *Peter E. Massar*

Developmental Editor: *Darlene M. Schueller*

Senior Marketing Manager: *Curt Reynolds*

Senior Project Manager: *Lisa A. Bruflodt*

Buyer: *Laura Fuller*

Design Coordinator: *Brenda A. Rolwes*

Media Project Manager: *Balaji Sundararaman*

Cover Design: *Studio Montage, St. Louis, Missouri*

Cover Image: © *Royalty-Free/CORBIS*

Compositor: *Techsetters, Inc.*

Typeface: *10/12 Times Roman*

Printer: *R. R. Donnelley & Sons Company/Crawfordsville, IN*

Library of Congress Cataloging-in-Publication Data

Computer organization and embedded systems / Carl Hamacher ... [et al.]. – 6th ed.
p. cm.

Includes bibliographical references.

ISBN-13: 978-0-07-338065-0 (alk. paper)

ISBN-10: 0-07-338065-2 (alk. paper)

1. Computer organization. 2. Embedded computer systems. I. Hamacher, V. Carl.

QA76.9.C643.H36 2012

004.2'2-dc22

2010050243

To our families

This page intentionally left blank

ABOUT THE AUTHORS

Carl Hamacher received the B.A.Sc. degree in Engineering Physics from the University of Waterloo, Canada, the M.Sc. degree in Electrical Engineering from Queen's University, Canada, and the Ph.D. degree in Electrical Engineering from Syracuse University, New York. From 1968 to 1990 he was at the University of Toronto, Canada, where he was a Professor in the Department of Electrical Engineering and the Department of Computer Science. He served as director of the Computer Systems Research Institute during 1984 to 1988, and as chairman of the Division of Engineering Science during 1988 to 1990. In 1991 he joined Queen's University, where is now Professor Emeritus in the Department of Electrical and Computer Engineering. He served as Dean of the Faculty of Applied Science from 1991 to 1996. During 1978 to 1979, he was a visiting scientist at the IBM Research Laboratory in San Jose, California. In 1986, he was a research visitor at the Laboratory for Circuits and Systems associated with the University of Grenoble, France. During 1996 to 1997, he was a visiting professor in the Computer Science Department at the University of California at Riverside and in the LIP6 Laboratory of the University of Paris VI.

His research interests are in multiprocessors and multicomputers, focusing on their interconnection networks.

Zvonko Vranesic received his B.A.Sc., M.A.Sc., and Ph.D. degrees, all in Electrical Engineering, from the University of Toronto. From 1963 to 1965 he worked as a design engineer with the Northern Electric Co. Ltd. in Bramalea, Ontario. In 1968 he joined the University of Toronto, where he is now a Professor Emeritus in the Department of Electrical & Computer Engineering. During the 1978–79 academic year, he was a Senior Visitor at the University of Cambridge, England, and during 1984–85 he was at the University of Paris, 6. From 1995 to 2000 he served as Chair of the Division of Engineering Science at the University of Toronto. He is also involved in research and development at the Altera Toronto Technology Center.

His current research interests include computer architecture and field-programmable VLSI technology.

He is a coauthor of four other books: *Fundamentals of Digital Logic with VHDL Design*, 3rd ed.; *Fundamentals of Digital Logic with Verilog Design*, 2nd ed.; *Microcomputer Structures*; and *Field-Programmable Gate Arrays*. In 1990, he received the Wighton Fellowship for “innovative and distinctive contributions to undergraduate laboratory instruction.” In 2004, he received the Faculty Teaching Award from the Faculty of Applied Science and Engineering at the University of Toronto.

Safwat Zaky received his B.Sc. degree in Electrical Engineering and B.Sc. in Mathematics, both from Cairo University, Egypt, and his M.A.Sc. and Ph.D. degrees in Electrical Engineering from the University of Toronto. From 1969 to 1972 he was with Bell Northern Research, Bramalea, Ontario, where he worked on applications of electro-optics and

This page intentionally left blank

magnetics in mass storage and telephone switching. In 1973, he joined the University of Toronto, where he is now Professor Emeritus in the Department of Electrical and Computer Engineering. He served as Chair of the Department from 1993 to 2003 and as Vice-Provost from 2003 to 2009. During 1980 to 1981, he was a senior visitor at the Computer Laboratory, University of Cambridge, England.

He is a Fellow of the Canadian Academy of Engineering. His research interests are in the areas of computer architecture, digital-circuit design, and electromagnetic compatibility. He is a coauthor of the book *Microcomputer Structures* and is a recipient of the IEEE Third Millennium Medal and of the Vivek Goel Award for distinguished service to the University of Toronto.

Naraig Manjikian received his B.A.Sc. degree in Computer Engineering and M.A.Sc. degree in Electrical Engineering from the University of Waterloo, Canada, and his Ph.D. degree in Electrical Engineering from the University of Toronto. In 1997, he joined Queen's University, Kingston, Canada, where he is now an Associate Professor in the Department of Electrical and Computer Engineering. From 2004 to 2006, he served as Undergraduate Chair for Computer Engineering. From 2006 to 2007, he served as Acting Head of the Department of Electrical and Computer Engineering, and from 2007 until 2009, he served as Associate Head for Student and Alumni Affairs. During 2003 to 2004, he was a visiting professor at McGill University, Montreal, Canada, and the University of British Columbia. During 2010 to 2011, he was a visiting professor at McGill University.

His research interests are in the areas of computer architecture, multiprocessor systems, field-programmable VLSI technology, and applications of parallel processing.

PREFACE

This book is intended for use in a first-level course on computer organization and embedded systems in electrical engineering, computer engineering, and computer science curricula. The book is self-contained, assuming only that the reader has a basic knowledge of computer programming in a high-level language. Many students who study computer organization will have had an introductory course on digital logic circuits. Therefore, this subject is not covered in the main body of the book. However, we have provided an extensive appendix on logic circuits for those students who need it.

The book reflects our experience in teaching three distinct groups of students: electrical and computer engineering undergraduates, computer science undergraduates, and engineering science undergraduates. We have always approached the teaching of courses on computer organization from a practical point of view. Thus, a key consideration in shaping the contents of the book has been to carefully explain the main principles, supported by examples drawn from commercially available processors. Our main commercial examples are based on: Altera's Nios II, Freescale's ColdFire, ARM, and Intel's IA-32 architectures.

It is important to recognize that digital system design is not a straightforward process of applying optimal design algorithms. Many design decisions are based largely on heuristic judgment and experience. They involve cost/performance and hardware/software tradeoffs over a range of alternatives. It is our goal to convey these notions to the reader.

The book is aimed at a one-semester course in engineering or computer science programs. It is suitable for both hardware- and software-oriented students. Even though the emphasis is on hardware, we have addressed a number of relevant software issues.

McGraw-Hill maintains a Website with support material for the book at <http://www.mhhe.com/hamacher>.

SCOPE OF THE BOOK

The first three chapters introduce the basic structure of computers, the operations that they perform at the machine-instruction level, and input/output methods as seen by a programmer. The fourth chapter provides an overview of the system software needed to translate programs written in assembly and high-level languages into machine language and to manage their execution. The remaining eight chapters deal with the organization, interconnection, and performance of hardware units in modern computers, including a coverage of embedded systems.

Five substantial appendices are provided. The first appendix covers digital logic circuits. Then, four current commercial instruction set architectures—Altera's Nios II, Freescale's ColdFire, ARM, and Intel's IA-32—are described in separate appendices.

Chapter 1 provides an overview of computer hardware and informally introduces terms that are discussed in more depth in the remainder of the book. This chapter discusses

the basic functional units and the ways they interact to form a complete computer system. Number and character representations are discussed, along with basic arithmetic operations. An introduction to performance issues and a brief treatment of the history of computer development are also provided.

Chapter 2 gives a methodical treatment of machine instructions, addressing techniques, and instruction sequencing. Program examples at the machine-instruction level, expressed in a generic assembly language, are used to discuss concepts that include loops, subroutines, and stacks. The concepts are introduced using a RISC-style instruction set architecture. A comparison with CISC-style instruction sets is also included.

Chapter 3 presents a programmer's view of basic input/output techniques. It explains how program-controlled I/O is performed using polling, as well as how interrupts are used in I/O transfers.

Chapter 4 considers system software. The tasks performed by compilers, assemblers, linkers, and loaders are explained. Utility programs that trace and display the results of executing a program are described. Operating system routines that manage the execution of user programs and their input/output operations, including the handling of interrupts, are also described.

Chapter 5 explores the design of a RISC-style processor. This chapter explains the sequence of processing steps needed to fetch and execute the different types of machine instructions. It then develops the hardware organization needed to implement these processing steps. The differing requirements of CISC-style processors are also considered.

Chapter 6 provides coverage of the use of pipelining and multiple execution units in the design of high-performance processors. A pipelined version of the RISC-style processor design from Chapter 5 is used to illustrate pipelining. The role of the compiler and the relationship between pipelined execution and instruction set design are explored. Superscalar processors are discussed.

Input/output hardware is considered in **Chapter 7**. Interconnection networks, including the bus structure, are discussed. Synchronous and asynchronous operation is explained. Interconnection standards, including USB and PCI Express, are also presented.

Semiconductor memories, including SDRAM, Rambus, and Flash memory implementations, are discussed in **Chapter 8**. Caches are explained as a way for increasing the memory bandwidth. They are discussed in some detail, including performance modeling. Virtual-memory systems, memory management, and rapid address-translation techniques are also presented. Magnetic and optical disks are discussed as components in the memory hierarchy.

Chapter 9 explores the implementation of the arithmetic unit of a computer. Logic design for fixed-point add, subtract, multiply, and divide hardware, operating on 2's-complement numbers, is described. Carry-lookahead adders and high-speed multipliers are explained, including descriptions of the Booth multiplier recoding and carry-save addition techniques. Floating-point number representation and operations, in the context of the IEEE Standard, are presented.

Today, far more processors are in use in embedded systems than in general-purpose computers. **Chapters 10 and 11** are dedicated to the subject of embedded systems. First, basic aspects of system integration, component interconnections, and real-time operation are presented in Chapter 10. The use of microcontrollers is discussed. Then, Chapter 11 concentrates on system-on-a-chip (SoC) implementations, in which a single chip integrates

the processing, memory, I/O, and timer functionality needed to satisfy application-specific requirements. A substantial example shows how FPGAs and modern design tools can be used in this environment.

Chapter 12 focuses on parallel processing and performance. Hardware multithreading and vector processing are introduced as enhancements in a single processor. Shared-memory multiprocessors are then described, along with the issue of cache coherence. Interconnection networks for multiprocessors are presented.

Appendix A provides extensive coverage of logic circuits, intended for a reader who has not taken a course on the design of such circuits.

Appendices B, C, D, and E illustrate how the instruction set concepts introduced in Chapters 2 and 3 are implemented in four commercial processors: Nios II, ColdFire, ARM, and Intel IA-32. The Nios II and ARM processors illustrate the RISC design style. ColdFire has an easy-to-teach CISC design, while the IA-32 CISC architecture represents the most successful commercial design. The presentation for each processor includes assembly-language examples from Chapters 2 and 3, implemented in the context of that processor. The details given in these appendices are not essential for understanding the material in the main body of the book. It is sufficient to cover only one of these appendices to gain an appreciation for commercial processor instruction sets. The choice of a processor to use as an example is likely to be influenced by the equipment in an accompanying laboratory. Instructors may wish to use more than one processor to illustrate the different design approaches.

CHANGES IN THE SIXTH EDITION

Substantial changes in content and organization have been made in preparing the sixth edition of this book. They include the following:

- The basic concepts of instruction set architecture are now covered using the RISC-style approach. This is followed by a comparative examination of the CISC-style approach.
- The processor design discussion is focused on a RISC-style implementation, which leads naturally to pipelined operation.
- Two chapters on embedded systems are included: one dealing with the basic structure of such systems and the use of microcontrollers, and the other dealing with system-on-a-chip implementations.
- Appendices are used to give examples of four commercial processors. Each appendix includes the essential information about the instruction set architecture of the given processor.
- Solved problems have been included in a new section toward the end of chapters and appendices. They provide the student with solutions that can be expected for typical problems.

DIFFICULTY LEVEL OF PROBLEMS

The problems at the end of chapters and appendices have been classified as easy (E), medium (M), or difficult (D). These classifications should be interpreted as follows:

- Easy—Solutions can be derived in a few minutes by direct application of specific information presented in one place in the relevant section of the book.
- Medium—Use of the book material in a way that does not directly follow any examples presented is usually needed. In some cases, solutions may follow the general pattern of an example, but will take longer to develop than those for easy problems.
- Difficult—Some additional insight is needed to solve these problems. If a solution requires a program to be written, its underlying algorithm or form may be quite different from that of any program example given in the book. If a hardware design is required, it may involve an arrangement and interconnection of basic logic circuit components that is quite different from any design shown in the book. If a performance analysis is needed, it may involve the derivation of an algebraic expression.

WHAT CAN BE COVERED IN A ONE-SEMESTER COURSE

This book is suitable for use at the university or college level as a text for a one-semester course in computer organization. It is intended for the first course that students will take on computer organization.

There is more than enough material in the book for a one-semester course. The core material on computer organization and relevant software issues is given in Chapters 1 through 9. For students who have not had a course in logic circuits, the material in Appendix A should be studied near the beginning of a course and certainly prior to covering Chapter 5.

A course aimed at embedded systems should include Chapters 1, 2, 3, 4, 7, 8, 10 and 11.

Use of the material on commercial processor examples in Appendices B through E can be guided by instructor and student interest, as well as by relevance to any hardware laboratory associated with a course.

ACKNOWLEDGMENTS

We wish to express our thanks to many people who have helped us during the preparation of this sixth edition of the book.

Our colleagues Daniel Etienne of University of Paris South and Glenn Gulak of University of Toronto provided numerous comments and suggestions that helped significantly in shaping the material.

Blair Fort and Dan Vranesic provided valuable help with some of the programming examples.

Warren R. Carithers of Rochester Institute of Technology, Krishna M. Kavi of University of North Texas, and Nelson Luiz Passos of Midwestern State University provided reviews of material from both the fifth and sixth editions of the book.

The following people provided reviews of material from the fifth edition of the book: Goh Hock Ann of Multimedia University, Joseph E. Beaini of University of Colorado Denver, Kalyan Mohan Goli of Jawaharlal Nehru Technological University, Jaimon Jacob of Model Engineering College Ernakulam, M. Kumaresan of Anna University Coimbatore,

Kenneth K. C. Lee of City University of Hong Kong, Manoj Kumar Mishra of Institute of Technical Education and Research, Junita Mohamad-Saleh of Universiti Sains Malaysia, Prashanta Kumar Patra of College of Engineering and Technology Bhubaneswar, Shanq-Jang Ruan of National Taiwan University of Science and Technology, S. D. Samantaray of G. B. Pant University of Agriculture and Technology, Shivakumar Sastry of University of Akron, Donatella Sciuto of Politecnico of Milano, M. P. Singh of National Institute of Technology Patna, Albert Starling of University of Arkansas, Shannon Tauro of University of California Irvine, R. Thangarajan of Kongu Engineering College, Ashok Kumar Turuk of National Institute of Technology Rourkela, and Philip A. Wilsey of University of Cincinnati.

Finally, we truly appreciate the support of Raghothaman Srinivasan, Peter E. Massar, Darlene M. Schueller, Lisa Bruflodt, Curt Reynolds, Brenda Rolwes, and Laura Fuller at McGraw-Hill.

Carl Hamacher
Zvonko Vranesic
Safwat Zaky
Naraig Manjikian

McGraw-Hill Create™ Craft your teaching resources to match the way you teach! With McGraw-Hill Create, www.mcgrawhillcreate.com, you can easily rearrange chapters, combine material from other content sources, and quickly upload content you have written like your course syllabus or teaching notes. Find the content you need in Create by searching through thousands of leading McGraw-Hill textbooks. Arrange your book to fit your teaching style. Create even allows you to personalize your book's appearance by selecting the cover and adding your name, school, and course information. Order a Create book and you'll receive a complimentary print review copy in 3-5 business days or a complimentary electronic review copy (eComp) via email in minutes. Go to www.mcgrawhillcreate.com today and register to experience how McGraw-Hill Create empowers you to teach your students your way.



McGraw-Hill Higher Education and Blackboard® have teamed up.

Blackboard, the Web-based course management system, has partnered with McGraw-Hill to better allow students and faculty to use online materials and activities to complement face-to-face teaching. Blackboard features exciting social learning and teaching tools that foster more logical, visually impactful and active learning opportunities for students. You'll transform your closed-door classrooms into communities where students remain connected to their educational experience 24 hours a day.

This partnership allows you and your students access to McGraw-Hill's Create right from within your Blackboard course - all with one single sign-on. McGraw-Hill and Blackboard can now offer you easy access to industry leading technology and content, whether your campus hosts it, or we do. Be sure to ask your local McGraw-Hill representative for details.

CONTENTS

Chapter 1

BASIC STRUCTURE OF COMPUTERS 1

- 1.1 Computer Types 2
- 1.2 Functional Units 3
 - 1.2.1 Input Unit 4
 - 1.2.2 Memory Unit 4
 - 1.2.3 Arithmetic and Logic Unit 5
 - 1.2.4 Output Unit 6
 - 1.2.5 Control Unit 6
- 1.3 Basic Operational Concepts 7
- 1.4 Number Representation and Arithmetic Operations 9
 - 1.4.1 Integers 10
 - 1.4.2 Floating-Point Numbers 16
- 1.5 Character Representation 17
- 1.6 Performance 17
 - 1.6.1 Technology 17
 - 1.6.2 Parallelism 19
- 1.7 Historical Perspective 19
 - 1.7.1 The First Generation 20
 - 1.7.2 The Second Generation 20
 - 1.7.3 The Third Generation 21
 - 1.7.4 The Fourth Generation 21
- 1.8 Concluding Remarks 22
- 1.9 Solved Problems 22
 - Problems 24
 - References 25

Chapter 2

INSTRUCTION SET ARCHITECTURE 27

- 2.1 Memory Locations and Addresses 28
 - 2.1.1 Byte Addressability 30
 - 2.1.2 Big-Endian and Little-Endian Assignments 30
 - 2.1.3 Word Alignment 31
 - 2.1.4 Accessing Numbers and Characters 32
- 2.2 Memory Operations 32

- 2.3 Instructions and Instruction Sequencing 32
 - 2.3.1 Register Transfer Notation 33
 - 2.3.2 Assembly-Language Notation 33
 - 2.3.3 RISC and CISC Instruction Sets 34
 - 2.3.4 Introduction to RISC Instruction Sets 34
 - 2.3.5 Instruction Execution and Straight-Line Sequencing 36
 - 2.3.6 Branching 37
 - 2.3.7 Generating Memory Addresses 40
- 2.4 Addressing Modes 40
 - 2.4.1 Implementation of Variables and Constants 41
 - 2.4.2 Indirection and Pointers 42
 - 2.4.3 Indexing and Arrays 45
- 2.5 Assembly Language 48
 - 2.5.1 Assembler Directives 50
 - 2.5.2 Assembly and Execution of Programs 53
 - 2.5.3 Number Notation 54
- 2.6 Stacks 55
- 2.7 Subroutines 56
 - 2.7.1 Subroutine Nesting and the Processor Stack 58
 - 2.7.2 Parameter Passing 59
 - 2.7.3 The Stack Frame 63
- 2.8 Additional Instructions 65
 - 2.8.1 Logic Instructions 67
 - 2.8.2 Shift and Rotate Instructions 68
 - 2.8.3 Multiplication and Division 71
- 2.9 Dealing with 32-Bit Immediate Values 73
- 2.10 CISC Instruction Sets 74
 - 2.10.1 Additional Addressing Modes 75
 - 2.10.2 Condition Codes 77
- 2.11 RISC and CISC Styles 78
- 2.12 Example Programs 79
 - 2.12.1 Vector Dot Product Program 79
 - 2.12.2 String Search Program 81
- 2.13 Encoding of Machine Instructions 82
- 2.14 Concluding Remarks 85
- 2.15 Solved Problems 85
 - Problems 90

Chapter 3

BASIC INPUT/OUTPUT 95

- 3.1 Accessing I/O Devices 96
 - 3.1.1 I/O Device Interface 97
 - 3.1.2 Program-Controlled I/O 97
 - 3.1.3 An Example of a RISC-Style I/O Program 101
 - 3.1.4 An Example of a CISC-Style I/O Program 101
- 3.2 Interrupts 103
 - 3.2.1 Enabling and Disabling Interrupts 106
 - 3.2.2 Handling Multiple Devices 107
 - 3.2.3 Controlling I/O Device Behavior 109
 - 3.2.4 Processor Control Registers 110
 - 3.2.5 Examples of Interrupt Programs 111
 - 3.2.6 Exceptions 116
- 3.3 Concluding Remarks 119
- 3.4 Solved Problems 119
 - Problems 126

Chapter 4

SOFTWARE 129

- 4.1 The Assembly Process 130
 - 4.1.1 Two-pass Assembler 131
- 4.2 Loading and Executing Object Programs 131
- 4.3 The Linker 132
- 4.4 Libraries 133
- 4.5 The Compiler 133
 - 4.5.1 Compiler Optimizations 134
 - 4.5.2 Combining Programs Written in Different Languages 134
- 4.6 The Debugger 134
- 4.7 Using a High-level Language for I/O Tasks 137
- 4.8 Interaction between Assembly Language and C Language 139
- 4.9 The Operating System 143
 - 4.9.1 The Boot-strapping Process 144
 - 4.9.2 Managing the Execution of Application Programs 144
 - 4.9.3 Use of Interrupts in Operating Systems 146
- 4.10 Concluding Remarks 149
 - Problems 149
 - References 150

Chapter 5

BASIC PROCESSING UNIT 151

- 5.1 Some Fundamental Concepts 152
- 5.2 Instruction Execution 155
 - 5.2.1 Load Instructions 155
 - 5.2.2 Arithmetic and Logic Instructions 156
 - 5.2.3 Store Instructions 157
- 5.3 Hardware Components 158
 - 5.3.1 Register File 158
 - 5.3.2 ALU 160
 - 5.3.3 Datapath 161
 - 5.3.4 Instruction Fetch Section 164
- 5.4 Instruction Fetch and Execution Steps 165
 - 5.4.1 Branching 168
 - 5.4.2 Waiting for Memory 171
- 5.5 Control Signals 172
- 5.6 Hardwired Control 175
 - 5.6.1 Datapath Control Signals 177
 - 5.6.2 Dealing with Memory Delay 177
- 5.7 CISC-Style Processors 178
 - 5.7.1 An Interconnect using Buses 180
 - 5.7.2 Microprogrammed Control 183
- 5.8 Concluding Remarks 185
- 5.9 Solved Problems 185
 - Problems 188

Chapter 6

PIPELINING 193

- 6.1 Basic Concept—The Ideal Case 194
- 6.2 Pipeline Organization 195
- 6.3 Pipelining Issues 196
- 6.4 Data Dependencies 197
 - 6.4.1 Operand Forwarding 198
 - 6.4.2 Handling Data Dependencies in Software 199
- 6.5 Memory Delays 201
- 6.6 Branch Delays 202
 - 6.6.1 Unconditional Branches 202
 - 6.6.2 Conditional Branches 204
 - 6.6.3 The Branch Delay Slot 204
 - 6.6.4 Branch Prediction 205
- 6.7 Resource Limitations 209
- 6.8 Performance Evaluation 209
 - 6.8.1 Effects of Stalls and Penalties 210
 - 6.8.2 Number of Pipeline Stages 212

- 6.9 Superscalar Operation 212
 - 6.9.1 Branches and Data Dependencies 214
 - 6.9.2 Out-of-Order Execution 215
 - 6.9.3 Execution Completion 216
 - 6.9.4 Dispatch Operation 217
- 6.10 Pipelining in CISC Processors 218
 - 6.10.1 Pipelining in ColdFire Processors 219
 - 6.10.2 Pipelining in Intel Processors 219
- 6.11 Concluding Remarks 220
- 6.12 Examples of Solved Problems 220
 - Problems 222
 - References 226

Chapter 7

INPUT/OUTPUT ORGANIZATION 227

- 7.1 Bus Structure 228
- 7.2 Bus Operation 229
 - 7.2.1 Synchronous Bus 230
 - 7.2.2 Asynchronous Bus 233
 - 7.2.3 Electrical Considerations 236
- 7.3 Arbitration 237
- 7.4 Interface Circuits 238
 - 7.4.1 Parallel Interface 239
 - 7.4.2 Serial Interface 243
- 7.5 Interconnection Standards 247
 - 7.5.1 Universal Serial Bus (USB) 247
 - 7.5.2 FireWire 251
 - 7.5.3 PCI Bus 252
 - 7.5.4 SCSI Bus 256
 - 7.5.5 SATA 258
 - 7.5.6 SAS 258
 - 7.5.7 PCI Express 258
- 7.6 Concluding Remarks 260
- 7.7 Solved Problems 260
 - Problems 263
 - References 266

Chapter 8

THE MEMORY SYSTEM 267

- 8.1 Basic Concepts 268
- 8.2 Semiconductor RAM Memories 270
 - 8.2.1 Internal Organization of Memory Chips 270
 - 8.2.2 Static Memories 271
 - 8.2.3 Dynamic RAMs 274

- 8.2.4 Synchronous DRAMs 276
- 8.2.5 Structure of Larger Memories 279
- 8.3 Read-only Memories 282
 - 8.3.1 ROM 283
 - 8.3.2 PROM 283
 - 8.3.3 EPROM 284
 - 8.3.4 EEPROM 284
 - 8.3.5 Flash Memory 284
- 8.4 Direct Memory Access 285
- 8.5 Memory Hierarchy 288
- 8.6 Cache Memories 289
 - 8.6.1 Mapping Functions 291
 - 8.6.2 Replacement Algorithms 296
 - 8.6.3 Examples of Mapping Techniques 297
- 8.7 Performance Considerations 300
 - 8.7.1 Hit Rate and Miss Penalty 301
 - 8.7.2 Caches on the Processor Chip 302
 - 8.7.3 Other Enhancements 303
- 8.8 Virtual Memory 305
 - 8.8.1 Address Translation 306
- 8.9 Memory Management Requirements 310
- 8.10 Secondary Storage 311
 - 8.10.1 Magnetic Hard Disks 311
 - 8.10.2 Optical Disks 317
 - 8.10.3 Magnetic Tape Systems 322
- 8.11 Concluding Remarks 323
- 8.12 Solved Problems 324
 - Problems 328
 - References 332

Chapter 9

ARITHMETIC 335

- 9.1 Addition and Subtraction of Signed Numbers 336
 - 9.1.1 Addition/Subtraction Logic Unit 336
- 9.2 Design of Fast Adders 339
 - 9.2.1 Carry-Lookahead Addition 340
- 9.3 Multiplication of Unsigned Numbers 344
 - 9.3.1 Array Multiplier 344
 - 9.3.2 Sequential Circuit Multiplier 346
- 9.4 Multiplication of Signed Numbers 346
 - 9.4.1 The Booth Algorithm 348
- 9.5 Fast Multiplication 351
 - 9.5.1 Bit-Pair Recoding of Multipliers 352
 - 9.5.2 Carry-Save Addition of Summands 353

9.5.3	Summand Addition Tree using 3-2 Reducers	355
9.5.4	Summand Addition Tree using 4-2 Reducers	357
9.5.5	Summary of Fast Multiplication	359
9.6	Integer Division	360
9.7	Floating-Point Numbers and Operations	363
9.7.1	Arithmetic Operations on Floating-Point Numbers	367
9.7.2	Guard Bits and Truncation	368
9.7.3	Implementing Floating-Point Operations	369
9.8	Decimal-to-Binary Conversion	372
9.9	Concluding Remarks	372
9.10	Solved Problems	374
	Problems	377
	References	383

Chapter 10

EMBEDDED SYSTEMS 385

10.1	Examples of Embedded Systems	386
10.1.1	Microwave Oven	386
10.1.2	Digital Camera	387
10.1.3	Home Telemetry	390
10.2	Microcontroller Chips for Embedded Applications	390
10.3	A Simple Microcontroller	392
10.3.1	Parallel I/O Interface	392
10.3.2	Serial I/O Interface	395
10.3.3	Counter/Timer	397
10.3.4	Interrupt-Control Mechanism	399
10.3.5	Programming Examples	399
10.4	Reaction Timer—A Complete Example	401
10.5	Sensors and Actuators	407
10.5.1	Sensors	407
10.5.2	Actuators	410
10.5.3	Application Examples	411
10.6	Microcontroller Families	412
10.6.1	Microcontrollers Based on the Intel 8051	413
10.6.2	Freescall Microcontrollers	413
10.6.3	ARM Microcontrollers	414
10.7	Design Issues	414
10.8	Concluding Remarks	417
	Problems	418
	References	420

Chapter 11

SYSTEM-ON-A-CHIP—A CASE STUDY 421

11.1	FPGA Implementation	422
11.1.1	FPGA Devices	423
11.1.2	Processor Choice	423
11.2	Computer-Aided Design Tools	424
11.2.1	Altera CAD Tools	425
11.3	Alarm Clock Example	428
11.3.1	User's View of the System	428
11.3.2	System Definition and Generation	429
11.3.3	Circuit Implementation	430
11.3.4	Application Software	431
11.4	Concluding Remarks	440
	Problems	440
	References	441

Chapter 12

PARALLEL PROCESSING AND PERFORMANCE 443

12.1	Hardware Multithreading	444
12.2	Vector (SIMD) Processing	445
12.2.1	Graphics Processing Units (GPUs)	448
12.3	Shared-Memory Multiprocessors	448
12.3.1	Interconnection Networks	450
12.4	Cache Coherence	453
12.4.1	Write-Through Protocol	453
12.4.2	Write-Back protocol	454
12.4.3	Snoopy Caches	454
12.4.4	Directory-Based Cache Coherence	456
12.5	Message-Passing Multicomputers	456
12.6	Parallel Programming for Multiprocessors	456
12.7	Performance Modeling	460
12.8	Concluding Remarks	461
	Problems	462
	References	463

Appendix A

LOGIC CIRCUITS 465

A.1	Basic Logic Functions	
A.1.1	Electronic Logic Gates	469
A.2	Synthesis of Logic Functions	470

A.3	Minimization of Logic Expressions	472
A.3.1	Minimization using Karnaugh Maps	475
A.3.2	Don't-Care Conditions	477
A.4	Synthesis with NAND and NOR Gates	479
A.5	Practical Implementation of Logic Gates	482
A.5.1	CMOS Circuits	484
A.5.2	Propagation Delay	489
A.5.3	Fan-In and Fan-Out Constraints	490
A.5.4	Tri-State Buffers	491
A.6	Flip-Flops	492
A.6.1	Gated Latches	493
A.6.2	Master-Slave Flip-Flop	495
A.6.3	Edge Triggering	498
A.6.4	T Flip-Flop	498
A.6.5	JK Flip-Flop	499
A.6.6	Flip-Flops with Preset and Clear	501
A.7	Registers and Shift Registers	502
A.8	Counters	503
A.9	Decoders	505
A.10	Multiplexers	506
A.11	Programmable Logic Devices (PLDs)	509
A.11.1	Programmable Logic Array (PLA)	509
A.11.2	Programmable Array Logic (PAL)	511
A.11.3	Complex Programmable Logic Devices (CPLDs)	512
A.12	Field-Programmable Gate Arrays	514
A.13	Sequential Circuits	516
A.13.1	Design of an Up/Down Counter as a Sequential Circuit	516
A.13.2	Timing Diagrams	519
A.13.3	The Finite State Machine Model	520
A.13.4	Synthesis of Finite State Machines	521
A.14	Concluding Remarks	522
	Problems	522
	References	528

Appendix B

THE ALTERA NIOS II PROCESSOR 529

B.1	Nios II Characteristics	530
B.2	General-Purpose Registers	531
B.3	Addressing Modes	532
B.4	Instructions	533
B.4.1	Notation	533
B.4.2	Load and Store Instructions	534
B.4.3	Arithmetic Instructions	536

B.4.4	Logic Instructions	537
B.4.5	Move Instructions	537
B.4.6	Branch and Jump Instructions	538
B.4.7	Subroutine Linkage Instructions	541
B.4.8	Comparison Instructions	545
B.4.9	Shift Instructions	546
B.4.10	Rotate Instructions	547
B.4.11	Control Instructions	548
B.5	Pseudoinstructions	548
B.6	Assembler Directives	549
B.7	Carry and Overflow Detection	551
B.8	Example Programs	553
B.9	Control Registers	553
B.10	Input/Output	555
B.10.1	Program-Controlled I/O	556
B.10.2	Interrupts and Exceptions	556
B.11	Advanced Configurations of Nios II Processor	562
B.11.1	External Interrupt Controller	562
B.11.2	Memory Management Unit	562
B.11.3	Floating-Point Hardware	562
B.12	Concluding Remarks	563
B.13	Solved Problems	563
	Problems	568

Appendix C

THE COLD FIRE PROCESSOR 571

C.1	Memory Organization	572
C.2	Registers	572
C.3	Instructions	573
C.3.1	Addressing Modes	575
C.3.2	Move Instruction	577
C.3.3	Arithmetic Instructions	578
C.3.4	Branch and Jump Instructions	582
C.3.5	Logic Instructions	585
C.3.6	Shift Instructions	586
C.3.7	Subroutine Linkage Instructions	587
C.4	Assembler Directives	593
C.5	Example Programs	594
C.5.1	Vector Dot Product Program	594
C.5.2	String Search Program	595
C.6	Mode of Operation and Other Control Features	596
C.7	Input/Output	597
C.8	Floating-Point Operations	599
C.8.1	FMOVE Instruction	599

C.8.2	Floating-Point Arithmetic Instructions	600
C.8.3	Comparison and Branch Instructions	601
C.8.4	Additional Floating-Point Instructions	601
C.8.5	Example Floating-Point Program	602
C.9	Concluding Remarks	603
C.10	Solved Problems	603
	Problems	608
	References	609

Appendix D

THE ARM PROCESSOR 611

D.1	ARM Characteristics	612
D.1.1	Unusual Aspects of the ARM Architecture	612
D.2	Register Structure	613
D.3	Addressing Modes	614
D.3.1	Basic Indexed Addressing Mode	614
D.3.2	Relative Addressing Mode	615
D.3.3	Index Modes with Writeback	616
D.3.4	Offset Determination	616
D.3.5	Register, Immediate, and Absolute Addressing Modes	618
D.3.6	Addressing Mode Examples	618
D.4	Instructions	621
D.4.1	Load and Store Instructions	621
D.4.2	Arithmetic Instructions	622
D.4.3	Move Instructions	625
D.4.4	Logic and Test Instructions	626
D.4.5	Compare Instructions	627
D.4.6	Setting Condition Code Flags	628
D.4.7	Branch Instructions	628
D.4.8	Subroutine Linkage Instructions	631
D.5	Assembly Language	635
D.5.1	Pseudoinstructions	637
D.6	Example Programs	638
D.6.1	Vector Dot Product	639
D.6.2	String Search	639
D.7	Operating Modes and Exceptions	639
D.7.1	Banked Registers	641
D.7.2	Exception Types	642
D.7.3	System Mode	644
D.7.4	Handling Exceptions	644
D.8	Input/Output	646
D.8.1	Program-Controlled I/O	646
D.8.2	Interrupt-Driven I/O	648

D.9	Conditional Execution of Instructions	648
D.10	Coprocessors	650
D.11	Embedded Applications and the Thumb ISA	651
D.12	Concluding Remarks	651
D.13	Solved Problems	652
	Problems	657
	References	660

Appendix E

THE INTEL IA-32 ARCHITECTURE 661

E.1	Memory Organization	662
E.2	Register Structure	662
E.3	Addressing Modes	665
E.4	Instructions	668
E.4.1	Machine Instruction Format	670
E.4.2	Assembly-Language Notation	670
E.4.3	Move Instruction	671
E.4.4	Load-Effective-Address Instruction	671
E.4.5	Arithmetic Instructions	672
E.4.6	Jump and Loop Instructions	674
E.4.7	Logic Instructions	677
E.4.8	Shift and Rotate Instructions	678
E.4.9	Subroutine Linkage Instructions	679
E.4.10	Operations on Large Numbers	681
E.5	Assembler Directives	685
E.6	Example Programs	686
E.6.1	Vector Dot Product Program	686
E.6.2	String Search Program	686
E.7	Interrupts and Exceptions	687
E.8	Input/Output Examples	689
E.9	Scalar Floating-Point Operations	690
E.9.1	Load and Store Instructions	692
E.9.2	Arithmetic Instructions	693
E.9.3	Comparison Instructions	694
E.9.4	Additional Instructions	694
E.9.5	Example Floating-Point Program	694
E.10	Multimedia Extension (MMX) Operations	695
E.11	Vector (SIMD) Floating-Point Operations	696
E.12	Examples of Solved Problems	697
E.13	Concluding Remarks	702
	Problems	702
	References	703