

COMPUTER ORGANIZATION

FIFTH EDITION

Carl Hamacher

Queen's University

Zvonko Vranesic

University of Toronto

Safwat Zaky

University of Toronto



Boston Burr Ridge, IL Dubuque, IA Madison, WI New York San Francisco St. Louis
Bangkok Bogotá Caracas Kuala Lumpur Lisbon London Madrid Mexico City
Milan Montreal New Delhi Santiago Seoul Singapore Sydney Taipei Toronto

S I T Library
Valachil, Mangalore



Accn No: M000535

The McGraw-Hill Companies



COMPUTER ORGANIZATION, Fifth Edition
International Edition 2002

Exclusive rights by McGraw-Hill Education (Asia), for manufacture and export. This book cannot be re-exported from the country to which it is sold by McGraw-Hill. The International Edition is not available in North America.

Published by McGraw-Hill, a business unit of The McGraw-Hill Companies, Inc. 1221 Avenue of the Americas, New York, NY 10020. Copyright © 2002, 1996, 1990, 1990, 1984, 1978 by The McGraw-Hill Companies, Inc. All rights reserved. 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.

Srinivas Institute of Technology

20 19 18 17 16 15 14 13 12 11
20 09 08
CTF BJE

Acc. No: 535

Call No:

Library of Congress Control Number: 2001030712

When ordering this title, use ISBN 007-120411-3

Printed in Singapore

www.mhhe.com

ABOUT THE AUTHORS

Carl Hamacher received his B.A.Sc. degree in engineering physics from the University of Waterloo, Canada, an M.Sc. degree in electrical engineering from Queen's University, Kingston, Canada, and a Ph.D. degree in electrical engineering from Syracuse University, New York. From 1968 to 1990 he was at the University of Toronto, where he was a Professor in the Departments of Electrical Engineering and 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. Since January 1991 he has been a Professor of Electrical and Computer Engineering at Queen's University. He served as the 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 in France. In 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, France.

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, 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 in the Department of Electrical and Computer Engineering and the Department of Computer Science. During 1978 to 1979, he was a senior visitor at the University of Cambridge, England, and during 1984 to 1985 he was at the University of Paris VI, France. In 2000 to 2001, he was a principal software engineer at Altera Corporation in Toronto. From 1995 to 2000, he served as chair of the Division of Engineering Science at the University of Toronto.

His current research interests include computer architecture, field-programmable VLSI technology, and multiple-valued logic systems. He is a coauthor of three other books: *Fundamentals of Digital Logic with VHDL Design*, *Microcomputer Structures*, and *Field-Programmable Gate Arrays*. In 1990, he received the Wighton Fellowship for "innovative and distinctive contributions to undergraduate laboratory instruction."

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 magnetics in mass storage and telephone switching. In 1973, he joined the University of Toronto, where he is now a Professor in the Department of Electrical and Computer Engineering and the Department of Computer Science. Presently, he

ABOUT THE AUTHORS

serves as chair of the Department of Electrical and Computer Engineering. From 1980 to 1981, he was a senior visitor at the Computer Laboratory, University of Cambridge, England.

His research interests are in the areas of computer architecture, reliability of digital circuits, and electromagnetic compatibility. He is a coauthor of the book *Microcomputer Structures* and is a recipient of the IEEE Third Millennium Medal.

To Liz, Anne, and Shirley

CONTENTS

Preface xvii

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 Bus Structures 9
- 1.5 Software 10
- 1.6 Performance 13
 - 1.6.1 Processor Clock 14
 - 1.6.2 Basic Performance Equation 14
 - 1.6.3 Pipelining and Superscalar Operation 15
 - 1.6.4 Clock Rate 16
 - 1.6.5 Instruction Set: CISC and RISC 16
 - 1.6.6 Compiler 17
 - 1.6.7 Performance Measurement 17
- 1.7 Multiprocessors and Multicomputers 18
- 1.8 Historical Perspective 19
 - 1.8.1 The First Generation 19
 - 1.8.2 The Second Generation 20
 - 1.8.3 The Third Generation 20
 - 1.8.4 The Fourth Generation 20
 - 1.8.5 Beyond the Fourth Generation 21
 - 1.8.6 Evolution of Performance 21
- 1.9 Concluding Remarks 21
 - Problems 22
 - References 23

Chapter 2

MACHINE INSTRUCTIONS AND PROGRAMS 25

- 2.1 Numbers, Arithmetic Operations, and Characters 27

- 2.1.1 Number Representation 27
- 2.1.2 Addition of Positive Numbers 28
- 2.1.3 Addition and Subtraction of Signed Numbers 29
- 2.1.4 Overflow in Integer Arithmetic 32
- 2.1.5 Characters 33
- 2.2 Memory Locations and Addresses 33
 - 2.2.1 Byte Addressability 33
 - 2.2.2 Big-endian and Little-endian Assignments 35
 - 2.2.3 Word Alignment 36
 - 2.2.4 Accessing Numbers, Characters, and Character Strings 36
- 2.3 Memory Operations 36
- 2.4 Instructions and Instruction Sequencing 37
 - 2.4.1 Register Transfer Notation 37
 - 2.4.2 Assembly Language Notation 38
 - 2.4.3 Basic Instruction Types 38
 - 2.4.4 Instruction Execution and Straight-Line Sequencing 42
 - 2.4.5 Branching 44
 - 2.4.6 Condition Codes 46
 - 2.4.7 Generating Memory Addresses 47
- 2.5 Addressing Modes 48
 - 2.5.1 Implementation of Variables and Constants 49
 - 2.5.2 Indirection and Pointers 50
 - 2.5.3 Indexing and Arrays 52
 - 2.5.4 Relative Addressing 56
 - 2.5.5 Additional Modes 56
- 2.6 Assembly Language 58
 - 2.6.1 Assembler Directives 59
 - 2.6.2 Assembly and Execution of Programs 62
 - 2.6.3 Number Notation 64
- 2.7 Basic Input/Output Operations 64
- 2.8 Stacks and Queues 68
- 2.9 Subroutines 72
 - 2.9.1 Subroutine Nesting and the Processor Stack 73
 - 2.9.2 Parameter Passing 74
 - 2.9.3 The Stack Frame 75
- 2.10 Additional Instructions 81

- 2.10.1 Logic Instructions 81
- 2.10.2 Shift and Rotate Instructions 82
- 2.10.3 Multiplication and Division 86
- 2.11 Example Programs 86
 - 2.11.1 Vector Dot Product Program 86
 - 2.11.2 Byte-Sorting Program 87
 - 2.11.3 Linked Lists 89
- 2.12 Encoding of Machine Instructions 94
- 2.13 Concluding Remarks 98
 - Problems 98

Chapter 3

ARM, MOTOROLA, AND INTEL INSTRUCTION SETS 103

Part I The ARM Example 104

- 3.1 Registers, Memory Access, and Data Transfer 104
 - 3.1.1 Register Structure 105
 - 3.1.2 Memory Access Instructions and Addressing Modes 106
 - 3.1.3 Register Move Instructions 113
- 3.2 Arithmetic and Logic Instructions 113
 - 3.2.1 Arithmetic Instructions 113
 - 3.2.2 Logic Instructions 115
- 3.3 Branch Instructions 116
 - 3.3.1 Setting Condition Codes 117
 - 3.3.2 A Loop Program for Adding Numbers 118
- 3.4 Assembly Language 118
 - 3.4.1 Pseudo-Instructions 120
- 3.5 I/O Operations 121
- 3.6 Subroutines 122
- 3.7 Program Examples 126
 - 3.7.1 Vector Dot Product Program 126
 - 3.7.2 Byte-Sorting Program 127
 - 3.7.3 Linked-List Insertion and Deletion Subroutines 127

Part II The 68000 Example 130

- 3.8 Registers and Addressing 131
 - 3.8.1 The 68000 Register Structure 131
 - 3.8.2 Addressing 131
- 3.9 Instructions 136
- 3.10 Assembly Language 140
- 3.11 Program Flow Control 141

- 3.11.1 Condition Code Flags 141
- 3.11.2 Branch Instructions 141
- 3.12 I/O Operations 145
- 3.13 Stacks and Subroutines 146
- 3.14 Logic Instructions 151
- 3.15 Program Examples 152
 - 3.15.1 Vector Dot Product Program 152
 - 3.15.2 Byte-Sorting Program 153
 - 3.15.3 Linked-List Insertion and Deletion Subroutines 154

Part III The IA-32 Pentium Example 155

- 3.16 Registers and Addressing 156
 - 3.16.1 IA-32 Register Structure 156
 - 3.16.2 IA-32 Addressing Modes 159
- 3.17 IA-32 Instructions 164
 - 3.17.1 Machine Instruction Format 168
- 3.18 IA-32 Assembly Language 170
- 3.19 Program Flow Control 171
 - 3.19.1 Conditional Jumps and Condition Code Flags 171
 - 3.19.2 Unconditional Jump 173
- 3.20 Logic and Shift/Rotate Instructions 173
 - 3.20.1 Logic Operations 173
 - 3.20.2 Shift and Rotate Operations 173
- 3.21 I/O Operations 174
 - 3.21.1 Memory-Mapped I/O 174
 - 3.21.2 Isolated I/O 175
 - 3.21.3 Block Transfers 176
- 3.22 Subroutines 177
- 3.23 Other Instructions 182
 - 3.23.1 Multiply and Divide Instructions 182
 - 3.23.2 Multimedia Extension (MMX) Instructions 183
 - 3.23.3 Vector (SIMD) Instructions 184
- 3.24 Program Examples 184
 - 3.24.1 Vector Dot Product Program 184
 - 3.24.2 Byte-Sorting Program 185
 - 3.24.3 Linked-List Insertion and Deletion Subroutines 185
- 3.25 Concluding Remarks 188
 - Problems 188
 - References 201

Chapter 4

INPUT/OUTPUT ORGANIZATION 203

- 4.1 Accessing I/O Devices 204
- 4.2 Interrupts 208

- 6.7.4 Implementing Floating-Point Operations 400
- 6.8 Concluding Remarks 403
- Problems 403
- References 410

Chapter 7

BASIC PROCESSING UNIT 411

- 7.1 Some Fundamental Concepts 412
 - 7.1.1 Register Transfers 415
 - 7.1.2 Performing an Arithmetic or Logic Operation 415
 - 7.1.3 Fetching a Word from Memory 418
 - 7.1.4 Storing a Word in Memory 420
- 7.2 Execution of a Complete Instruction 421
 - 7.2.1 Branch Instructions 422
- 7.3 Multiple-Bus Organization 423
- 7.4 Hardwired Control 425
 - 7.4.1 A Complete Processor 428
- 7.5 Microprogrammed Control 429
 - 7.5.1 Microinstructions 432
 - 7.5.2 Microprogram Sequencing 435
 - 7.5.3 Wide-Branch Addressing 437
 - 7.5.4 Microinstructions with Next-Address Field 440
 - 7.5.5 Prefetching Microinstructions 443
 - 7.5.6 Emulation 443
- 7.6 Concluding Remarks 445
 - Problems 446

Chapter 8

PIPELINING 453

- 8.1 Basic Concepts 454
 - 8.1.1 Role of Cache Memory 456
 - 8.1.2 Pipeline Performance 458
- 8.2 Data Hazards 461
 - 8.2.1 Operand Forwarding 462
 - 8.2.2 Handling Data Hazards in Software 464
 - 8.2.3 Side Effects 464
- 8.3 Instruction Hazards 465
 - 8.3.1 Unconditional Branches 466
 - 8.3.2 Conditional Branches and Branch Prediction 470
- 8.4 Influence on Instruction Sets 476
 - 8.4.1 Addressing Modes 476
 - 8.4.2 Condition Codes 478

- 8.5 Datapath and Control Considerations 479
- 8.6 Superscalar Operation 481
 - 8.6.1 Out-of-Order Execution 483
 - 8.6.2 Execution Completion 485
 - 8.6.3 Dispatch Operation 486
- 8.7 UltraSPARC II EXAMPLE 486
 - 8.7.1 SPARC Architecture 487
 - 8.7.2 UltraSPARC II 493
 - 8.7.3 Pipeline Structure 493
- 8.8 Performance Considerations 503
 - 8.8.1 Effect of Instruction Hazards 504
 - 8.8.2 Number of Pipeline Stages 505
- 8.9 Concluding Remarks 506
 - Problems 506
 - Reference 509

Chapter 9

EMBEDDED SYSTEMS 511

- 9.1 Examples of Embedded Systems 512
 - 9.1.1 Microwave Oven 512
 - 9.1.2 Digital Camera 514
 - 9.1.3 Home Telemetry 516
- 9.2 Processor Chips for Embedded Applications 517
- 9.3 A Simple Microcontroller 518
 - 9.3.1 Parallel I/O Ports 518
 - 9.3.2 Serial I/O Interface 521
 - 9.3.3 Counter/Timer 523
 - 9.3.4 Interrupt Control Mechanism 525
- 9.4 Programming Considerations 525
 - 9.4.1 Polling Approach 526
 - 9.4.2 Interrupt Approach 529
- 9.5 I/O Device Timing Constraints 531
 - 9.5.1 C Program for Transfer via a Circular Buffer 533
 - 9.5.2 Assembly Language Program for Transfer via a Circular Buffer 534
- 9.6 Reaction Timer — An Example 535
 - 9.6.1 C Program for the Reaction Timer 537
 - 9.6.2 Assembly Language Program for the Reaction Timer 537
 - 9.6.3 Final Comments 541
- 9.7 Embedded Processor Families 541
 - 9.7.1 Microcontrollers Based on the Intel 8051 542
 - 9.7.2 Motorola Microcontrollers 542
 - 9.7.3 ARM Microcontrollers 543

- 9.8 Design Issues 544
- 9.9 System-on-a-Chip 546
 - 9.9.1 FPGA Implementation 547
- 9.10 Concluding Remarks 549
 - Problems 550
 - References 552

Chapter 10

COMPUTER PERIPHERALS 553

- 10.1 Input Devices 554
 - 10.1.1 Keyboard 554
 - 10.1.2 Mouse 555
 - 10.1.3 Trackball, Joystick, and Touchpad 556
 - 10.1.4 Scanners 557
- 10.2 Output Devices 558
 - 10.2.1 Video Displays 558
 - 10.2.2 Flat-Panel Displays 559
 - 10.2.3 Printers 560
 - 10.2.4 Graphics Accelerators 561
- 10.3 Serial Communication Links 563
 - 10.3.1 Asynchronous Transmission 566
 - 10.3.2 Synchronous Transmission 568
 - 10.3.3 Standard Communications Interfaces 571
- 10.4 Concluding Remarks 574
 - Problems 575

Chapter 11

PROCESSOR FAMILIES 577

- 11.1 The ARM Family 579
 - 11.1.1 The Thumb Instruction Set 579
 - 11.1.2 Processor and CPU Cores 580
- 11.2 The Motorola 680X0 and ColdFire Families 582
 - 11.2.1 68020 Processor 582
 - 11.2.2 Enhancements in 68030 and 68040 Processors 584
 - 11.2.3 68060 Processor 585
 - 11.2.4 The ColdFire Family 585
- 11.3 The Intel IA-32 Family 585
 - 11.3.1 IA-32 Memory Segmentation 586
 - 11.3.2 Sixteen-Bit Mode 588
 - 11.3.3 80386 and 80486 Processors 588
 - 11.3.4 Pentium Processor 589
 - 11.3.5 Pentium Pro Processor 589
 - 11.3.6 Pentium II and III Processors 590

- 11.3.7 Pentium 4 Processor 590
- 11.3.8 Advanced Micro Devices IA-32 Processors 591
- 11.4 The PowerPC Family 591
 - 11.4.1 Register Set 591
 - 11.4.2 Memory Addressing Modes 592
 - 11.4.3 Instructions 592
 - 11.4.4 PowerPC Processors 592
- 11.5 The Sun Microsystems SPARC Family 594
- 11.6 The Compaq Alpha Family 596
 - 11.6.1 Instruction and Addressing Mode Formats 596
 - 11.6.2 Alpha 21064 Processor 597
 - 11.6.3 Alpha 21164 Processor 597
 - 11.6.4 Alpha 21264 Processor 597
- 11.7 The Intel IA-64 Family 598
 - 11.7.1 Instruction Bundles 598
 - 11.7.2 Conditional Execution 598
 - 11.7.3 Speculative Loads 600
 - 11.7.4 Registers and the Register Stack 600
 - 11.7.5 Itanium Processor 602
- 11.8 A Stack Processor 603
 - 11.8.1 Stack Structure 604
 - 11.8.2 Stack Instructions 606
 - 11.8.3 Hardware Registers in the Stack 610
- 11.9 Concluding Remarks 612
 - Problems 612
 - References 614

Chapter 12

LARGE COMPUTER SYSTEMS 617

- 12.1 Forms of Parallel Processing 619
 - 12.1.1 Classification of Parallel Structures 619
- 12.2 Array Processors 620
- 12.3 The Structure of General-Purpose Multiprocessors 622
- 12.4 Interconnection Networks 624
 - 12.4.1 Single Bus 624
 - 12.4.2 Crossbar Networks 625
 - 12.4.3 Multistage Networks 626
 - 12.4.4 Hypercube Networks 628
 - 12.4.5 Mesh Networks 630
 - 12.4.6 Tree Networks 630
 - 12.4.7 Ring Networks 631
 - 12.4.8 Practical Considerations 632
 - 12.4.9 Mixed Topology Networks 636
 - 12.4.10 Symmetric Multiprocessors 636

- 12.5 Memory Organization in Multiprocessors 637
 - 12.6 Program Parallelism and Shared Variables 638
 - 12.6.1 Accessing Shared Variables 640
 - 12.6.2 Cache Coherence 641
 - 12.6.3 Need for Locking and Cache Coherence 645
 - 12.7 Multicomputers 645
 - 12.7.1 Local Area Networks 646
 - 12.7.2 Ethernet (CSMA/CD) Bus 646
 - 12.7.3 Token Ring 647
 - 12.7.4 Network of Workstations 647
 - 12.8 Programmer's View of Shared Memory and Message Passing 648
 - 12.8.1 Shared Memory Case 648
 - 12.8.2 Message-Passing Case 651
 - 12.9 Performance Considerations 653
 - 12.9.1 Amdahl's Law 654
 - 12.9.2 Performance Indicators 656
 - 12.10 Concluding Remarks 656
 - Problems 657
 - References 660
- APPENDIX A: LOGIC CIRCUITS 661**
- A.1 Basic Logic Functions 662
 - A.1.1 Electronic Logic Gates 665
 - A.2 Synthesis of Logic Functions 666
 - A.3 Minimization of Logic Expressions 668
 - A.3.1 Minimization Using Karnaugh Maps 671
 - A.3.2 Don't-Care Conditions 674
 - A.4 Synthesis with NAND and NOR Gates 674
 - A.5 Practical Implementation of Logic Gates 678
 - A.5.1 CMOS Circuits 681
 - A.5.2 Propagation Delay 686
 - A.5.3 Fan-In and Fan-Out Constraints 687
 - A.5.4 Tri-state Buffers 687
 - A.5.5 Integrated Circuit Packages 688
 - A.6 Flip-Flops 690
 - A.6.1 Gated Latches 690
 - A.6.2 Master-Slave Flip-Flop 694
 - A.6.3 Edge Triggering 694
 - A.6.4 T Flip-Flop 697

- A.6.5 JK Flip-Flop 697
- A.6.6 Flip-Flops with Preset and Clear 698
- A.7 Registers and Shift Registers 699
- A.8 Counters 702
- A.9 Decoders 703
- A.10 Multiplexers 705
- A.11 Programmable Logic Devices (PLDs) 705
 - A.11.1 Programmable Logic Array (PLA) 707
 - A.11.2 Programmable Array Logic (PAL) 710
 - A.11.3 Complex Programmable Logic Devices (CPLDs) 711
- A.12 Field-Programmable Gate Arrays 712
- A.13 Sequential Circuits 714
 - A.13.1 An Example of an Up/Down Counter 714
 - A.13.2 Timing Diagrams 718
 - A.13.3 The Finite State Machine Model 719
 - A.13.4 Synthesis of Finite State Machines 720
- A.14 Concluding Remarks 724
 - Problems 724
 - References 731

APPENDIX B: ARM INSTRUCTION SET 733

- B.1 Instruction Encoding 734
 - B.1.1 Arithmetic and Logic Instructions 734
 - B.1.2 Memory Load and Store Instructions 741
 - B.1.3 Block Load and Store Instructions 744
 - B.1.4 Branch and Branch with Link Instructions 747
 - B.1.5 Machine Control Instructions 747
- B.2 Other ARM Instructions 750
 - B.2.1 Coprocessor Instructions 750
 - B.2.2 Versions v4 and v5 Instructions 750
- B.3 Programming Experiments 750

APPENDIX C: MOTOROLA 68000 INSTRUCTION SET 751

**APPENDIX D: INTEL IA-32
INSTRUCTION SET 769**

- D.1 Instruction Encoding 770
 - D.1.1 Addressing Modes 772
- D.2 Basic Instructions 773
 - D.2.1 Conditional Jump Instructions 782
 - D.2.2 Unconditional Jump Instructions 782
- D.3 Prefix Bytes 782
- D.4 Other Instructions 783
 - D.4.1 String Instructions 783

- D.4.2 Floating-Point, MMX, and SSE
Instructions 784
- D.5 Sixteen-Bit Operation 785
- D.6 Programming Experiments 785

**APPENDIX E: CHARACTER CODES AND
NUMBER CONVERSION 789**

- E.1 Character Codes 790
- E.2 Decimal-to-Binary Conversion 793

INDEX 795

PREFACE

This book is intended for use in a first-level course on computer organization 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 computer organization to three distinct groups of undergraduates: electrical and computer engineering undergraduates, computer science specialists, and engineering science undergraduates. We have always approached the teaching of courses in this area from a practical point of view. Thus, a key consideration in shaping the contents of the book has been to illustrate the principles of computer organization using examples drawn from commercially available computers. Our main examples are based on the following processors: ARM, Motorola 680X0, Intel Pentium, and Sun UltraSPARC.

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.

We have endeavored to provide sufficient details to encourage the student to dig beyond the surface when dealing with ideas that seem to be intuitively obvious. We believe that this is best accomplished by giving real examples that are adequately documented. Block diagrams are a powerful means of describing organizational features of a computer. However, they can easily lead to an oversimplified view of the problems involved. Hence, they must be accompanied by the details of implementation alternatives.

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 software issues, including basic aspects of compilers and operating systems related to instruction execution performance, coordination of parallel operations at the system level, and real-time applications. An understanding of hardware/software interaction and tradeoffs is necessary for computer specialists.

THE SCOPE OF THE BOOK

We now review the topics covered in sequence, chapter by chapter. The first eight chapters cover the basic principles of computer organization, operation, and performance.

The remaining four chapters deal with embedded systems, peripheral devices, processor family evolution patterns, and large computer systems.

Chapter 1 provides an overview of computer hardware and software and informally introduces terms that are dealt with in more depth in the remainder of the book. This chapter discusses the basic functional units and the ways they are interconnected to form a complete computer system. The role of system software is introduced and basic aspects of performance evaluation are discussed. A brief treatment of the history of computer development is also provided.

Chapter 2 gives a methodical treatment of machine instructions, addressing techniques, and instruction sequencing. Basic aspects of 2's-complement arithmetic are introduced to facilitate the discussion of the generation of effective addresses. Program examples at the machine instruction level, expressed in a generic assembly language, are used to discuss loops, subroutines, simple input-output programming, sorting, and linked list operations.

Chapter 3 illustrates implementation of the concepts introduced in Chapter 2 on three commercial processors — ARM, 68000, and Pentium. The ARM processor illustrates the RISC design style, the 68000 has an easy-to-teach CISC design, while the Pentium represents the most successful commercial design that combines the elements of both the CISC and RISC styles. The material is organized into three independent and complete parts. Each part includes all of the examples from Chapter 2 implemented in the context of the specific processor. It is sufficient to cover only one of the three parts to provide the continuity needed to follow the rest of the book. If laboratory experiments using one of the three processors are associated with the course, the relevant part of Chapter 3 can be covered in parallel with Chapter 2.

Input-output organization is developed in Chapter 4. The basics of I/O data transfer synchronization are presented, and a series of increasingly complex I/O structures are explained. Interrupts and direct-memory access methods are described in detail, including a discussion of the role of software interrupts in operating systems. Bus protocols and standards are also presented, with the PCI, SCSI, and USB standards being used as representative commercial examples.

Semiconductor memories, including SDRAM, Rambus, and Flash memory implementations, are discussed in Chapter 5. Caches and multiple-module memory systems are explained as ways for increasing main memory bandwidth. Caches 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 6 treats 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. 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.

Chapter 7 begins with a register-transfer-level treatment of the implementation of instruction fetching and execution in a processor. This is followed by a discussion of processor implementation by both hardwired and microprogrammed control.

Chapter 8 provides a detailed coverage of the use of pipelining and multiple function units in the design of high-performance processors. The role of the compiler and the relationship between pipelined execution and instruction set design are explored. Superscalar processors are discussed, and the Sun Microsystems UltraSPARC II processor organization is used to illustrate the concepts.

Today there are many more processors in use in embedded systems than in general-purpose computers. This increasingly important subject, where a single chip integrates the processing, I/O, and timer functionality needed in a wide range of low-cost applications, is treated in Chapter 9. System integration issues, interconnections, and real-time software are discussed.

Chapter 10 presents peripheral devices and computer interconnections. Typical input/output devices are described and hardware needed to support computer graphics applications is introduced. Commonly used communication links, such as DSL, are discussed.

The evolution of the ARM, Motorola, and Intel processor families is discussed in Chapter 11. This chapter highlights the design changes that led to higher performance. The PowerPC, SPARC, Alpha, and Intel IA-64 families are also discussed.

Chapter 12 extends the discussion of computer organization to large systems that use many processors operating in parallel. Interconnection networks for multiprocessors are described, and an introduction to cache coherence controls is presented. Shared-memory and message-passing schemes are discussed.

CHANGES IN THE FIFTH EDITION

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

- Chapter 2 of the fourth edition has been split into two chapters — Chapters 2 and 3 — in the fifth edition. An expanded treatment of basic issues, explained using generic instructions, is presented in Chapter 2. More programming examples for typical tasks, both numeric and non-numeric, are provided. Chapter 3 uses the instruction sets of ARM, 68000, and Pentium processors to show how the basic concepts of instruction set design have been implemented in both the RISC and CISC design styles.
- The discussion of the role of pipelining and multiple functional units in processor design has been extended significantly. The UltraSPARC architecture is used to provide specific examples of performance-enhancing design features.
- A new chapter on embedded-processor systems has been added. A generic design of a typical system is used as the basis for detailed discussion of example applications.

In addition to these main changes, many recent technology and design advances have been added to a number of chapters.

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 use in the first course on computer organization that the students will take.

There is more than enough material in the book for a one-semester course. The core material is given in Chapters 1 through 8. For students who have not had a course in logic circuits, the basic material in Appendix A should be studied at the beginning of the course and certainly prior to covering Chapter 4.

Chapters 9 through 12 contain a variety of useful material that the instructor may choose from if time permits. Particularly suitable are the discussion of embedded systems in Chapter 9 and the description of hardware found in most personal computers given in Chapter 10.

ACKNOWLEDGMENTS

We wish to express our thanks to many people who have helped during the preparation of this fifth edition. Gail Burgess and Kelly Chan helped with the technical preparation of the manuscript. Alex Grbic, Frank Hsu and Robert Lu provided valuable help with a number of programming examples. Our colleagues Tarek Abdelrahman, Stephen Brown, Paul Chow, Glenn Gulak and Jonathan Rose offered constructive comments. We are particularly grateful to Stephen and Tarek for their help with important details. The reviewers, Gojko Babic of The Ohio State University, Nathaniel Davis of Virginia Polytechnic Institute and State University, Jose Fortes of Purdue University, John Greiner of Rice University, Sung Hu of San Francisco State University, Ali Hurson of Pennsylvania State University, Lizy Kurian John of University of Texas at Austin, Stefan Leue of Albert Ludwigs Universitat in Freiburg, Fabrizio Lombardi of Northeastern University, Wayne Loucks of University of Waterloo, Prasant Mohapatra of Iowa State University, Daniel Tabak of George Mason University, and John Valois of Rensselaer Polytechnic Institute gave us many excellent suggestions and provided constructive criticism. We want to thank Eli Vranesic for permission to use his painting "Fall in High Park" on the front cover; he created it using the computer as a paint brush. Finally, we truly appreciate the support of our editor, Catherine Fields Shultz, and her McGraw-Hill associates: Kelley Butcher, Michelle Flomenhofs, Kalah Graham, Betsy Jones, Rick Noel, Heather Sabo, and Christine Walker.

Carl Hamacher
Zvonko Vranesic
Safwat Zaky