国外优秀信息科学与技术系列教学用书

计算机系统结构基础

（影印版）

ESSENTIALS OF COMPUTER ARCHITECTURE

■ Douglas E. Comer

PEARSON
Prentice Hall

高等教育出版社
Higher Education Press

国外优秀信息科学与技术系列教学用书

计算机系统结构基础

（影印版）

ESSENTIALS OF COMPUTER ARCHITECTURE

Douglas E. Comer

高等教育出版社

图字:01 -2005 -0964 号

Essentials of Computer Architecture

Douglas E. Comer

本书封面贴有 Pearson Education(培生教育出版集团)激光防伪标签。无标签者不得销售。

English reprint edition copyright © 2005 by **PEARSON EDUCATION ASIA LIMITIED** and **HIGHER EDUCATION PRESS.** (Essentials of Computer Architecture from Pearson Education's edition of the Work) ***Essentials of Computer Architecture***, *le* by *Douglas E. Comer*, Copyright © 2005.

All Rights Reserved.

Published by arrangement with the original publisher, Pearson Education, Inc., publishing as Pearson Education, Inc..

This edition is authorized for sale only in the People's Republic of China (excluding the Special Administrative Regions of Hong Kong and Macau).

原版 ISBN:0 - 13 - 149179 - 2

For sale and distribution in the People's Republic of China exclusively (except Taiwan, Hong Kong SAR and Macao SAR).

仅限于中华人民共和国境内(但不允许在中国香港、澳门特别行政区和中国台湾地区)销售发行。

图书在版编目(CIP)数据

计算机系统结构基础 = Essentials of Computer Architecture/(美)科默(Comer, D. E.).—影印本.
北京:高等教育出版社,2005.12
ISBN 7 - 04 - 017816 - 8

Ⅰ.计... Ⅱ.科... Ⅲ.计算机体系结构 - 高等学校 - 教材 - 英文 Ⅳ.TP303

中国版本图书馆 CIP 数据核字(2005)第 132966 号

出版发行	高等教育出版社	**购书热线**	010 - 58581118
社　　址	北京市西城区德外大街 4 号	**免费咨询**	800 - 810 - 0598
邮政编码	100011	**网　　址**	http://www.hep.edu.cn
总　　机	010 - 58581000		http://www.hep.com.cn
经　　销	北京蓝色畅想图书发行有限公司	**网上订购**	http://www.landraco.com
印　　刷	北京外文印刷厂		http://www.landraco.com.cn
开　　本	787×1092　1/16	**版　　次**	2005 年 12 月第 1 版
印　　张	25	**印　　次**	2005 年 12 月第 1 次印刷
字　　数	480 000	**定　　价**	38.00 元

本书如有缺页、倒页、脱页等质量问题,请到所购图书销售部门联系调换。

版权所有　侵权必究

物料号　17816 - 00

Preface

This book began when I was assigned to help salvage an undergraduate computer organization course. The course had suffered years of neglect: it had been taught by a series of professors, mostly visitors, who had little or no interest or background in digital hardware, and the curriculum had deteriorated to a potpourri of topics that were only loosely related to hardware architectures. In some semesters, students spent the entire class studying Boolean Algebra, without even the slightest connection to actual hardware. In others, students learned the arcane details of one particular assembly language, without a notion of alternatives.

Is a computer organization course worth saving? Absolutely! In many Computer Science programs, the computer organization course is the only time students are exposed to fundamental concepts that explain the structure of the computer they are programming. Understanding the hardware makes it possible to construct programs that are more efficient and less prone to errors. In a broad sense, a basic knowledge of architecture helps programmers improve program efficiency by understanding the consequences of programming choices. Knowing how the hardware works can also improve the programming process by allowing programmers to pinpoint the source of bugs quickly. Finally, graduates need to understand basic architectural concepts to pass job application tests given by firms like Intel and Microsoft.

One of the steps in salvaging our architecture course consisted in looking at textbooks. We discovered the texts could be divided into roughly two types: texts aimed at beginning engineering students who would go on to design hardware, and texts written for CS students that attempt to include topics from compilers, operating systems, and (in at least one case) a complete explanation of how Internet protocols operate. Neither approach seemed appropriate for a single, introductory course on the subject. We wanted a book that (1) focused on the concepts rather than engineering details (because our students are not focused on hardware design); (2) explained the subject from a programmer's point of view, and emphasized consequences for programmers; and (3) did not try to cover several courses' worth of material. When no text was found, it seemed that the only solution was to create one.

The text is divided into five parts. Part 1 covers the basics of digital logic, gates, and data representation. We emphasize the representation chapter because notions of two's-compliment arithmetic and ranges of integer values are essential in programming. Parts 2, 3, and 4 cover the three essential areas of architecture: processors, memories, and I/O systems. In each case, the chapters give students enough background to under-

stand how the mechanisms operate and the consequences for programmers. Finally, Part 5 covers advanced topics like parallelism, pipelining, and performance.

An Appendix describes an important aspect of the course: a hands-on lab where students can learn by doing. Although most lab problems focus on programming, students should spend the first few weeks in lab wiring a few gates on a breadboard. The equipment is inexpensive (we spent less than fifteen dollars per student on permanent equipment; students purchase their own set of chips for under twenty dollars).

We have set up a web site to accompany the book at:

http://www.eca.cs.purdue.edu

Rajesh Subraman has agreed to manage the site, which contains a set of class presentation materials created by the author as well as a set created by Rajesh. We invite other instructors to contribute their materials.

The text and lab exercises have been used at Purdue; students have been extremely positive about both. We received notes of thanks for the text and course. For many students, the lab is their first experience with hardware, and they are enthusiastic.

My thanks to the many individuals who contributed to the book. Bernd Wolfinger provided extensive reviews and made several important suggestions about topics and direction. Dan Ardelean, James Cernak, and Tim Korb gave detailed comments on many chapters. Dave Capka reviewed early chapters. Rajesh Subraman taught from the book and provided his thoughts about the content. In the CS 250 class at Purdue, the following students each identified one or more typos in the manuscript: Nitin Alreja, Alex Cox, David Ehrmann, Roger Maurice Elion, Andrew Lee, Stan Luban, Andrew L. Soderstrom, and Brandon Wuest.

Finally, I thank my wife, Chris, for her patient and careful editing and valuable suggestions that improve and polish each book.

Douglas E. Comer

June, 2004

试读结束：需要全本请在线购买： www.ertongbook.com

About The Author

Dr. Douglas Comer has an extensive background in computer systems, and has worked with both hardware and software. Comer's work on software spans most aspects of systems, including compilers and operating systems. He created a complete operating system, including a process manager, a memory manager, and device drivers for both serial and parallel interfaces. Comer has also implemented network protocol software and network device drivers for conventional computers and network processors. Both his operating system, Xinu, and TCP/IP protocol stack have been used in commercial products.

Comer's experience with hardware includes work with discrete components, building circuits from logic gates, and experience with basic silicon technology. He has written popular textbooks on network processor architectures, and at Bell Laboratories, Comer studied VLSI design and fabricated a VLSI chip.

Comer is a Distinguished Professor of Computer Science at Purdue University, where he develops and teaches courses and does research on computer organization, operating systems, networks, and Internets. Comer has created innovative laboratories in which students can build and measure systems such as operating systems and IP routers; all of Comer's courses include hands-on lab work. He continues to consult and lecture at universities, industries, and conferences around the world.

In addition to writing a series of internationally acclaimed technical books on computer operating systems, networks, TCP/IP, and computer technologies, Comer serves as the editor-in-chief of the journal *Software — Practice and Experience*. He is a Fellow of the ACM, a Fellow of the Purdue Teaching Academy, and a recipient of numerous awards, including a Usenix Lifetime Achievement award.

Additional information can be found at:

www.cs.purdue.edu/people/comer

and information about Comer's books can be found at:

www.comerbooks.com

Contents

Preface xxi

Chapter 1 Introduction And Overview 1

1.1 The Importance Of Architecture 1
1.2 Learning The Essentials 1
1.3 Organization Of The Text 2
1.4 What We Will Omit 3
1.5 Terminology: Architecture And Design 3
1.6 Summary 3

PART I Basics

Chapter 2 Fundamentals Of Digital Logic 7

2.1 Introduction 7
2.2 Electrical Terminology: Voltage And Current 7
2.3 The Transistor 8
2.4 Logic Gates 9
2.5 Symbols Used For Gates 10
2.6 Construction Of Gates From Transistors 11
2.7 Example Interconnection Of Gates 12
2.8 Multiple Gates Per Integrated Circuit 14
2.9 The Need For More Than Combinatorial Circuits 15
2.10 Circuits That Maintain State 15
2.11 Transition Diagrams 16
2.12 Binary Counters 17
2.13 Clocks And Sequences 18
2.14 The Important Concept Of Feedback 20
2.15 Starting A Sequence 22
2.16 Iteration In Software Vs. Replication In Hardware 22
2.17 Gate And Chip Minimization 23
2.18 Using Spare Gates 24

2.19 *Power Distribution And Heat Dissipation* 24
2.20 *Timing* 25
2.21 *Physical Size And Process Technologies* 26
2.22 *Circuit Boards And Layers* 27
2.23 *Levels Of Abstraction* 27
2.24 *Summary* 28

Chapter 3 Data And Program Representation 29

3.1 *Introduction* 29
3.2 *Digital Logic And Abstraction* 29
3.3 *Bits And Bytes* 30
3.4 *Byte Size And Possible Values* 30
3.5 *Binary Arithmetic* 31
3.6 *Hexadecimal Notation* 32
3.7 *Notation For Hexadecimal And Binary Constants* 33
3.8 *Character Sets* 34
3.9 *Unicode* 35
3.10 *Unsigned Integers, Overflow, And Underflow* 35
3.11 *Numbering Bits And Bytes* 36
3.12 *Signed Integers* 37
3.13 *An Example Of Two's Complement Numbers* 38
3.14 *Sign Extension* 39
3.15 *Floating Point* 40
3.16 *Special Values* 42
3.17 *Range Of IEEE Floating Point Values* 42
3.18 *Data Aggregates* 42
3.19 *Program Representation* 43
3.20 *Summary* 43

PART II Processors

Chapter 4 The Variety Of Processors And Computational Engines 47

4.1 *Introduction* 47
4.2 *Von Neumann Architecture* 47
4.3 *Definition Of A Processor* 48
4.4 *The Range Of Processors* 48
4.5 *Hierarchical Structure And Computational Engines* 49
4.6 *Structure Of A Conventional Processor* 51
4.7 *Definition Of An Arithmetic Logic Unit (ALU)* 52

4.8 Processor Categories And Roles 52
4.9 Processor Technologies 54
4.10 Stored Programs 54
4.11 The Fetch-Execute Cycle 55
4.12 Clock Rate And Instruction Rate 56
4.13 Control: Getting Started And Stopping 57
4.14 Starting The Fetch-Execute Cycle 57
4.15 Summary 58

Chapter 5 Processor Types And Instruction Sets 61

5.1 Introduction 61
5.2 Mathematical Power, Convenience, And Cost 61
5.3 Instruction Set And Representation 62
5.4 Opcodes, Operands, And Results 63
5.5 Typical Instruction Format 63
5.6 Variable-Length Vs. Fixed-Length Instructions 63
5.7 General-Purpose Registers 64
5.8 Floating Point Registers And Register Identification 65
5.9 Programming With Registers 65
5.10 Register Banks 66
5.11 Complex And Reduced Instruction Sets 67
5.12 RISC Design And The Execution Pipeline 68
5.13 Pipelines And Instruction Stalls 69
5.14 Other Causes Of Pipeline Stalls 71
5.15 Consequences For Programmers 71
5.16 Programming, Stalls, And No-Op Instructions 72
5.17 Forwarding 72
5.18 Types Of Operations 73
5.19 Program Counter, Fetch-Execute, And Branching 73
5.20 Subroutine Calls, Arguments, And Register Windows 75
5.21 An Example Instruction Set 76
5.22 Minimalistic Instruction Set 78
5.23 The Principle Of Orthogonality 79
5.24 Condition Codes And Conditional Branching 80
5.25 Summary 80

Chapter 6 Operand Addressing And Instruction Representation 83

6.1 Introduction 83
6.2 Zero, One, Two, Or Three Address Designs 83
6.3 Zero Operands Per Instruction 84

6.4 *One Operand Per Instruction* 85
6.5 *Two Operands Per Instruction* 85
6.6 *Three Operands Per Instruction* 86
6.7 *Operand Sources And Immediate Values* 86
6.8 *The Von Neumann Bottleneck* 87
6.9 *Explicit And Implicit Operand Encoding* 88
6.10 *Operands That Combine Multiple Values* 89
6.11 *Tradeoffs In The Choice Of Operands* 90
6.12 *Values In Memory And Indirect Reference* 91
6.13 *Operand Addressing Modes* 92
6.14 *Summary* 93

Chapter 7 CPUs: Microcode, Protection, And Processor Modes 95

7.1 *Introduction* 95
7.2 *A Central Processor* 95
7.3 *CPU Complexity* 96
7.4 *Modes Of Execution* 97
7.5 *Backward Compatibility* 97
7.6 *Changing Modes* 98
7.7 *Privilege And Protection* 99
7.8 *Multiple Levels Of Protection* 99
7.9 *Microcoded Instructions* 100
7.10 *Microcode Variations* 102
7.11 *The Advantage Of Microcode* 102
7.12 *Making Microcode Visible To Programmers* 103
7.13 *Vertical Microcode* 103
7.14 *Horizontal Microcode* 104
7.15 *Example Horizontal Microcode* 105
7.16 *A Horizontal Microcode Example* 107
7.17 *Operations That Require Multiple Cycles* 108
7.18 *Horizontal Microcode And Parallel Execution* 109
7.19 *Look-Ahead And High Performance Execution* 110
7.20 *Parallelism And Execution Order* 111
7.21 *Out-Of-Order Instruction Execution* 111
7.22 *Conditional Branches And Branch Prediction* 112
7.23 *Consequences For Programmers* 113
7.24 *Summary* 113

Chapter 8 Assembly Languages And Programming Paradigm 115

8.1 Introduction 115
8.2 Characteristics Of A High-level Programming Language 115
8.3 Characteristics Of A Low-Level Programming Language 116
8.4 Assembly Language 117
8.5 Assembly Language Syntax And Opcodes 118
8.6 Operand Order 120
8.7 Register Names 121
8.8 Operand Types 122
8.9 Assembly Language Programming Paradigm And Idioms 122
8.10 Assembly Code For Conditional Execution 123
8.11 Assembly Code For A Conditional Alternative 124
8.12 Assembly Code For Definite Iteration 124
8.13 Assembly Code For Indefinite Iteration 125
8.14 Assembly Code For Procedure Invocation 125
8.15 Assembly Code For Parameterized Procedure Invocation 126
8.16 Consequence For Programmers 127
8.17 Assembly Code For Function Invocation 128
8.18 Interaction Between Assembly And High-Level Languages 128
8.19 Assembly Code For Variables And Storage 129
8.20 Two-Pass Assembler 130
8.21 Assembly Language Macros 131
8.22 Summary 134

PART III Memories

Chapter 9 Memory And Storage 137

9.1 Introduction 137
9.2 Definition 137
9.3 The Key Aspects Of Memory 138
9.4 Characteristics Of Memory Technologies 138
9.5 The Important Concept Of A Memory Hierarchy 140
9.6 Instruction And Data Store 140
9.7 The Fetch-Store Paradigm 141
9.8 Summary 141

试读结束：需要全本请在线购买： www.ertongbook.com

Chapter 10 Physical Memory And Physical Addressing 143

10.1 Introduction 143
10.2 Characteristics Of Computer Memory 143
10.3 Static And Dynamic RAM Technologies 144
10.4 Measures Of Memory Technology 145
10.5 Density 146
10.6 Separation Of Read And Write Performance 146
10.7 Latency And Memory Controllers 146
10.8 Synchronized Memory Technologies 147
10.9 Multiple Data Rate Memory Technologies 148
10.10 Examples Of Memory Technologies 148
10.11 Memory Organization 148
10.12 Memory Access And Memory Bus 149
10.13 Memory Transfer Size 150
10.14 Physical Addresses And Words 150
10.15 Physical Memory Operations 150
10.16 Word Size And Other Data Types 151
10.17 An Extreme Case: Byte Addressing 151
10.18 Byte Addressing With Word Transfers 152
10.19 Using Powers Of Two 153
10.20 Byte Alignment And Programming 154
10.21 Memory Size And Address Space 154
10.22 Programming With Word Addressing 155
10.23 Measures Of Memory Size 155
10.24 Pointers And Data Structures 156
10.25 A Memory Dump 156
10.26 Indirection And Indirect Operands 158
10.27 Memory Banks And Interleaving 158
10.28 Content Addressable Memory 159
10.29 Ternary CAM 160
10.30 Summary 160

Chapter 11 Virtual Memory Technologies And Virtual Addressing 163

11.1 Introduction 163
11.2 Definition 163
11.3 A Virtual Example: Byte Addressing 164
11.4 Virtual Memory Terminology 164
11.5 An Interface To Multiple Physical Memory Systems 164
11.6 Address Translation Or Address Mapping 166
11.7 Avoiding Arithmetic Calculation 167
11.8 Discontiguous Address Spaces 168

11.9 Other Memory Organizations 169
11.10 Motivation For Virtual Memory 169
11.11 Multiple Virtual Spaces And Multiprogramming 170
11.12 Multiple Levels Of Virtualization 171
11.13 Creating Virtual Spaces Dynamically 171
11.14 Base-Bound Registers 172
11.15 Changing The Virtual Space 172
11.16 Virtual Memory, Base-Bound, And Protection 173
11.17 Segmentation 174
11.18 Demand Paging 175
11.19 Hardware And Software For Demand Paging 175
11.20 Page Replacement 176
11.21 Paging Terminology And Data Structures 176
11.22 Address Translation In A Paging System 177
11.23 Using Powers Of Two 178
11.24 Presence, Use, And Modified Bits 179
11.25 Page Table Storage 180
11.26 Paging Efficiency And A Translation Lookaside Buffer 181
11.27 Consequences For Programmers 182
11.28 Summary 183

Chapter 12 Caches And Caching 185

12.1 Introduction 185
12.2 Definition 185
12.3 Characteristics Of A Cache 186
12.4 The Importance Of Caching 187
12.5 Examples Of Caching 188
12.6 Cache Terminology 188
12.7 Best And Worst Case Cache Performance 189
12.8 Cache Performance On A Typical Sequence 190
12.9 Cache Replacement Policy 190
12.10 LRU Replacement 191
12.11 Multi-level Cache Hierarchy 191
12.12 Preloading Caches 192
12.13 Caches Used With Memory 192
12.14 TLB As A Cache 193
12.15 Demand Paging As A Form Of Caching 193
12.16 Physical Memory Cache 194
12.17 Write Through And Write Back 194
12.18 Cache Coherence 195
12.19 L1, L2, and L3 Caches 196
12.20 Sizes Of L1, L2, And L3 Caches 197

12.21 Instruction And Data Caches 197
12.22 Virtual Memory Caching And A Cache Flush 198
12.23 Implementation Of Memory Caching 199
12.24 Direct Mapping Memory Cache 200
12.25 Using Powers Of Two For Efficiency 201
12.26 Set Associative Memory Cache 202
12.27 Consequences For Programmers 203
12.28 Summary 204

PART IV I/O

Chapter 13 Input/Output Concepts And Terminology 207

13.1 Introduction 207
13.2 Input And Output Devices 207
13.3 Control Of An External Device 208
13.4 Data Transfer 209
13.5 Serial And Parallel Data Transfers 209
13.6 Self-Clocking Data 210
13.7 Full-Duplex And Half-Duplex Interaction 210
13.8 Interface Latency And Throughput 211
13.9 The Fundamental Idea Of Multiplexing 211
13.10 Multiple Devices Per External Interface 212
13.11 A Processor's View Of I/O 213
13.12 Summary 213

Chapter 14 Buses And Bus Architectures 215

14.1 Introduction 215
14.2 Definition Of A Bus 215
14.3 Processors, I/O Devices, And Buses 216
14.4 Proprietary And Standardized Buses 216
14.5 Shared Buses And An Access Protocol 217
14.6 Multiple Buses 217
14.7 A Parallel, Passive Mechanism 217
14.8 Physical Connections 217
14.9 Bus Interface 218
14.10 Address, Control, And Data Lines 219
14.11 The Fetch-Store Paradigm 220
14.12 Fetch-Store Over A Bus 220
14.13 The Width Of A Bus 220

14.14 Multiplexing 221
14.15 Bus Width And Size Of Data Items 222
14.16 Bus Address Space 223
14.17 Potential Errors 224
14.18 Address Configuration And Sockets 225
14.19 Many Buses Or One Bus 226
14.20 Using Fetch-Store With Devices 226
14.21 An Example Of Device Control Using Fetch-Store 226
14.22 Operation Of An Interface 227
14.23 Asymmetric Assignments 228
14.24 Unified Memory And Device Addressing 228
14.25 Holes In The Address Space 230
14.26 Address Map 230
14.27 Program Interface To A Bus 231
14.28 Bridging Between Two Buses 232
14.29 Main And Auxiliary Buses 232
14.30 Consequences For Programmers 234
14.31 Switching Fabrics 234
14.32 Summary 235

Chapter 15 Programmed And Interrupt-Driven I/O 237

15.1 Introduction 237
15.2 I/O Paradigms 237
15.3 Programmed I/O 238
15.4 Synchronization 238
15.5 Polling 239
15.6 Code For Polling 239
15.7 Control And Status Registers 241
15.8 Processor Use And Polling 241
15.9 First, Second, And Third Generation Computers 242
15.10 Interrupt-Driven I/O 242
15.11 A Hardware Interrupt Mechanism 243
15.12 Interrupts And The Fetch-Execute Cycle 243
15.13 Handling An Interrupt 244
15.14 Interrupt Vectors 245
15.15 Initialization And Enabling And Disabling Interrupts 246
15.16 Preventing Interrupt Code From Being Interrupted 246
15.17 Multiple Levels Of Interrupts 246
15.18 Assignment Of Interrupt Vectors And Priorities 247
15.19 Dynamic Bus Connections And Pluggable Devices 248
15.20 The Advantage Of Interrupts 249
15.21 Smart Devices And Improved I/O Performance 249

15.22 Direct Memory Access (DMA) 250
15.23 Buffer Chaining 251
15.24 Scatter Read And Gather Write Operations 252
15.25 Operation Chaining 252
15.26 Summary 253

Chapter 16 A Programmer's View Of Devices, I/O, And Buffering 255

16.1 Introduction 255
16.2 Definition Of A Device Driver 256
16.3 Device Independence, Encapsulation, And Hiding 256
16.4 Conceptual Parts Of A Device Driver 257
16.5 Two Types Of Devices 258
16.6 Example Flow Through A Device Driver 258
16.7 Queued Output Operations 259
16.8 Forcing An Interrupt 261
16.9 Queued Input Operations 261
16.10 Devices That Support Bi-Directional Transfer 262
16.11 Asynchronous Vs. Synchronous Programming Paradigm 263
16.12 Asynchrony, Smart Devices, And Mutual Exclusion 264
16.13 I/O As Viewed By An Application 264
16.14 Run-Time I/O Libraries 265
16.15 The Library/Operating System Dichotomy 266
16.16 I/O Operations The OS Supports 267
16.17 The Cost Of I/O Operations 268
16.18 Reducing The System Call Overhead 268
16.19 The Important Concept Of Buffering 269
16.20 Implementation of Buffering 270
16.21 Flushing A Buffer 271
16.22 Buffering On Input 272
16.23 Effectiveness Of Buffering 272
16.24 Buffering In An Operating System 273
16.25 Relation To Caching 274
16.26 An Example: The Unix Standard I/O Library 274
16.27 Summary 274

PART V Advanced Topics

Chapter 17 Parallelism 279

17.1 Introduction 279
17.2 Parallel And Pipelined Architectures 279
17.3 Characterizations Of Parallelism 280
17.4 Microscopic Vs. Macroscopic 280
17.5 Examples Of Microscopic Parallelism 281
17.6 Examples Of Macroscopic Parallelism 281
17.7 Symmetric Vs. Asymmetric 282
17.8 Fine-grain Vs. Coarse-grain Parallelism 282
17.9 Explicit Vs. Implicit Parallelism 283
17.10 Parallel Architectures 283
17.11 Types Of Parallel Architectures (Flynn Classification) 283
17.12 Single Instruction Single Data (SISD) 284
17.13 Single Instruction Multiple Data (SIMD) 284
17.14 Multiple Instructions Multiple Data (MIMD) 286
17.15 Communication, Coordination, And Contention 288
17.16 Performance Of Multiprocessors 290
17.17 Consequences For Programmers 292
17.18 Redundant Parallel Architectures 294
17.19 Distributed And Cluster Computers 295
17.20 Summary 296

Chapter 18 Pipelining 299

18.1 Introduction 299
18.2 The Concept Of Pipelining 299
18.3 Software Pipelining 301
18.4 Software Pipeline Performance And Overhead 302
18.5 Hardware Pipelining 303
18.6 How Hardware Pipelining Increases Performance 303
18.7 When Pipelining Can Be Used 306
18.8 The Conceptual Division Of Processing 307
18.9 Pipeline Architectures 307
18.10 Pipeline Setup, Stall, And Flush Times 308
18.11 Definition Of Superpipeline Architecture 308
18.12 Summary 309

试读结束：需要全本请在线购买： www.ertongbook.com