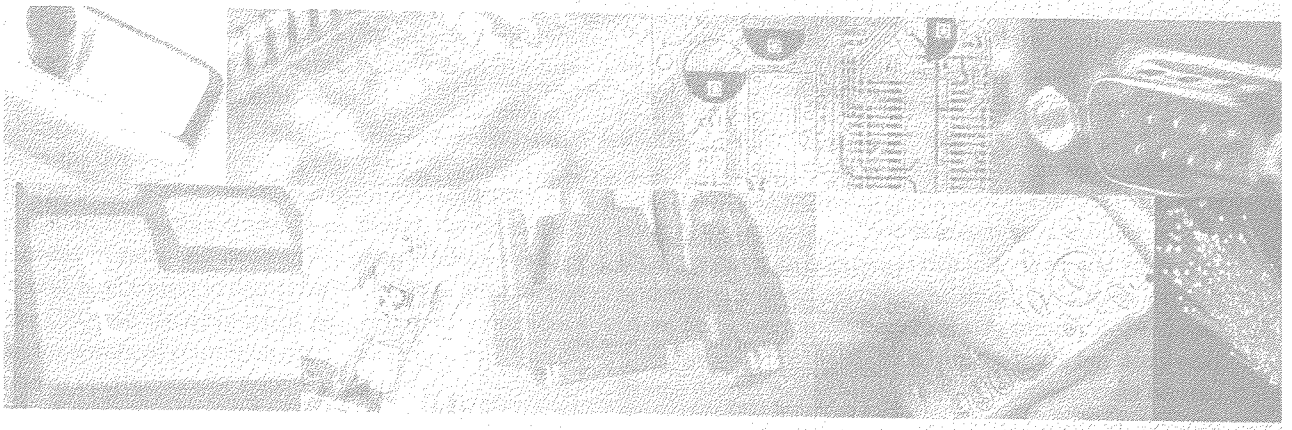


PRINCIPLES OF
**COMPUTER
HARDWARE**



Alan Clements

School of Computing
University of Teesside

Fourth Edition

OXFORD
UNIVERSITY PRESS

PREFACE

Principle of Computer Hardware is aimed at students taking an introductory course in electronics, computer science, or information technology. The approach is one of *breadth before depth* and we cover a wide range of topics under the general umbrella of *computer hardware*.

I have written *Principles of Computer Hardware* to achieve two goals. The first is to teach students the basic concepts on which the stored-program digital computer is founded. These include the representation and manipulation of information in binary form, the structure or *architecture* of a computer, the flow of information within a computer, and the exchange of information between its various peripherals. We answer the questions, 'How does a computer work', and 'How is it organized?' The second goal is to provide students with a foundation for further study. In particular, the elementary treatment of gates and Boolean algebra provides a basis for a second-level course in digital design, and the introduction to the CPU and assembly-language programming provides a basis for advanced courses on computer architecture/organization or microprocessor systems design.

This book is written for those with no previous knowledge of computer architecture. The only background information needed by the reader is an understanding of elementary algebra. Because students following a course in computer science or computer technology will also be studying a high-level language, we assume that the reader is familiar with the concepts underlying a high-level language.

When writing this book, I set myself three objectives. By adopting an informal style, I hope to increase the enthusiasm of students who may be put off by the formal approach of more traditional books. I have also tried to give students an insight into computer hardware by explaining why things are as they are, instead of presenting them with information to be learned and accepted without question. I have included subjects that would seem out of place in an elementary first-level

course. Topics like advanced computer arithmetic, timing diagrams, and reliability have been included to show how the computer hardware of the real world often differs from that of the first-level course in which only the basics are taught. I've also broadened the range of topics normally found in first-level courses in computer hardware and provided sections introducing operating systems and local area networks, as these two topics are so intimately related to the hardware of the computer. Finally, I have discovered that stating a formula or a theory is not enough—many students like to see an actual application of the formula. Wherever possible I have provided examples.

Like most introductory books on computer architecture, I have chosen a specific microprocessor as a vehicle to illustrate some of the important concepts in computer architecture. The ideal computer architecture is rich in features and yet easy to understand without exposing the student to a steep learning curve. Some microprocessors have very complicated architectures that confront the students with too much fine detail early in their course. We use Motorola's 68K microprocessor because it is easy to understand and incorporates many of the most important features of a high-performance architecture. This book isn't designed to provide a practical assembly language programming course. It is intended only to illustrate the operation of a central processing unit by means of a typical assembly language. We also take a brief look at other microprocessors to show the range of computer architectures available.

You will see the words *computer*, *CPU*, *processor*, *microprocessor*, and *microcomputer* in this and other texts. The part of a computer that actually executes a program is called a CPU (central processing unit) or more simply a *processor*. A *microprocessor* is a CPU fabricated on a single chip of silicon. A computer that is constructed around a *microprocessor* can be called a *microcomputer*. To a certain extent, these terms are frequently used interchangeably.

READING GUIDE

We've already said that this book provides a traditional introductory course in computer architecture plus additional material to broaden its scope and fill in some of the gaps left in such courses. To help students distinguish between foreground and background material, the following guide will help to indicate the more fundamental components of the course.

Chapter 2 introduces the logic of computers and deals with essential topics such as gates, Boolean algebra, and Karnaugh maps. Therefore this chapter is essential reading.

Chapter 3 introduces *sequential circuits* such as the counter that steps through the instructions of a program and demonstrates how sequential circuits are designed. We first introduce the *bistable* (flip-flop) used to construct *sequential circuits* such as registers and counters. We don't provide a comprehensive introduction to the design of sequential circuits; we show how gates and flip-flops can be used to create a computer.

Chapter 4 deals with the representation of numbers and shows how arithmetic operations are implemented. Apart from some of the coding theory and details of multiplication and division, almost all this chapter is essential reading. Multiplication and division can be omitted if the student is not interested in how these operations are implemented.

Chapter 5 is the heart of the book and is concerned with the structure and operation of the computer itself. We examine the instruction set of a processor with a sophisticated architecture.

Chapter 6 provides an overview of assembly language programming and the design of simple 68K assembly language programs. This chapter relies heavily on the 68K cross-assembler and simulator provided with the book. You can use this software to investigate the behavior of the 68K on a PC.

Chapter 7 begins with a description of the functional units that make up a computer and the flow of data during the execution of an instruction. We then describe the operation of the computer's control unit, which decodes and executes instructions. The control unit may be omitted on a first reading. Although the control unit is normally encountered in a second- or third-level course, we've included it here for the purpose of completeness and to show how the computer turns a binary-coded instruction into the sequence of events that carry out the instruction.

Chapter 8 is concerned with the quest for performance. We look at how performance is measured and describe three techniques used to accelerate processors. All students should read about the first two acceleration techniques, pipelining and cache memory, but may omit parallel processing.

Chapter 9 describes two contrasting computer architectures. Introductory texts on computer architecture are forced to concentrate on one processor because students do not have the time to plow through several different instruction sets. However, if we don't cover other architectures, students can end the course with a rather unbalanced view of processors. In this chapter we provide a very brief overview of several contrasting processors. We do not expect students to learn the fine details of these processors. The purpose of this chapter is to expose students to the range of processors that are available to the designer.

Chapter 10 deals with input/output techniques. We are interested in the way in which information is transferred between a computer and peripherals. We also examine the buses, or data highways, along which data flows. This chapter is essential reading.

Chapter 11 introduces some of the basic peripherals you'd find in a typical PC such as the keyboard, display, printer, and mouse, as well as some of the more unusual peripherals that, for example, can measure how fast a body is rotating. Although these topics are often omitted from courses in computer hardware, students should scan this chapter to get some insight into how computers control the outside world.

Chapter 12 looks at the memory devices used to store data in a computer. Information isn't stored in a computer in just one type of storage device. It's stored in DRAM and on disk, CD-ROM, DVD, and tape. This chapter examines the operating principles and characteristics of the storage devices found in a computer. There's a lot of detail in this chapter. Some readers may wish to omit the design of memory systems (for example, *address decoding and interfacing*) and just concentrate on the reasons why computers have so many different types of memory.

Chapter 13 deals with hardware topics that are closely related to the computer's operating system. The two most important elements of a computer's hardware that concern the operating system are *multiprogramming* and *memory management*. These topics are intimately connected with interrupt handling

and data storage techniques and serve as practical examples of the use of the hardware described elsewhere. Those who require a basic introduction to computer hardware may omit this chapter, although it best illustrates how hardware and software come together in the operating system.

Chapter 14 describes how computers can communicate with each other. The techniques used to link computers to create

computer networks are not always covered by first-level texts on computer architecture. However, the growth of both local area networks and the Internet have propelled computer communications to the forefront of computing. For this reason we would expect students to read this chapter even if some of it falls outside the scope of their syllabus.

THE HISTORY OF THIS BOOK

Like people, books are born. *Principles of Computer Hardware* was conceived in December 1980. At the end of their first semester our freshmen were given tests to monitor their progress. The results of the test in my ‘Principles of computer hardware’ course were not as good as I’d hoped, so I decided to do something about it. I thought that detailed lecture notes written in a style accessible to the students would be the most effective solution.

Having volunteered to give a course on computer communications to the staff of the Computer Center during the Christmas vacation, I didn’t have enough free time to produce the notes. By accident I found that the week before Christmas was the cheapest time of the year for vacations. So I went to one of the Canary Islands for a week, sat down by the pool, surrounded by folders full of reference material, with a bottle of Southern Comfort, and wrote the core of this book—number bases, gates, Boolean algebra, and binary arithmetic. Shortly afterwards I added the section on the structure of the CPU.

These notes produced the desired improvement in the end-of-semester exam results and were well received by the students. In the next academic year my notes were transferred from paper to a mainframe computer and edited to include new material and to clean up the existing text.

I decided to convert the notes into a book. The conversion process involved adding topics, not covered by our syllabus, to produce a more rounded text. While editing my notes, I discovered what might best be called the *inkblot effect*. Text stored in a computer tends to expand in all directions because it’s so easy to add new material at any point; for example, you might write a section on disk drives. When you next edit the section on disks, you can add more depth or breadth.

The final form of this book took a *breadth before depth* approach. That is, I covered a large number of topics rather than treating fewer topics in greater depth. It was my intention to give students taking our introductory hardware/architecture course a reasonably complete picture of the computer *system*.

The first edition of *Principles of Computer Hardware* proved successful and I was asked to write a second edition, which was published in 1990. The major change between the first and second editions was the adoption of the 68K microprocessors as a vehicle to teach computer architecture. I have retained this processor in the current edition. Although members of the Intel family have become the standard

processors in the PC world, Motorola’s 68K family of microprocessors is much better suited to teaching computer architecture. In short, it supports most of the features that computer scientists wish to teach students, and just as importantly, it’s much easier to understand. The 68K family and its derivatives are widely used in embedded systems.

By the mid-1990s the second edition was showing its age. The basic computer science and the underlying principles were still fine, but the actual hardware had changed dramatically over a very short time. The most spectacular progress was in the capacity of hard disks—by the late 1990s disk capacity was increasing by 60% per year.

This third edition included a 68K cross-assembler and simulator allowing students to create and run 68K programs on any PC. It also added details of interesting microprocessor architecture, the ARM, which provides an interesting contrast to the 68K.

When I used the second edition to teach logic design to my students, they built simple circuits using *logic trainers*—boxes with power supplies and connectors that allow you to wire a handful of simple chips together. Dave Barker, one of my former students, has constructed a logic simulator program as part of his senior year project called Digital Works, which runs under Windows on a PC. Digital Works allows you to place logic elements anywhere within a window and to wire the gates together. Inputs to the gates can be provided manually (via the mouse) or from clocks and sequence generators. You can observe the outputs of the gates on synthesized LEDs or as a waveform or table. Moreover, Digital Works permits you to encapsulate a circuit in a macro and then use this macro in other circuits. In other words, you can take gates and build simple circuits, and take the simple circuits and build complex circuits, and so on.

I began writing a fourth edition of this text in late 2003. The fundamental principles have changed little since the third edition, but processors had become faster by a factor of 10 and the capacity of hard disks has grown enormously. This new edition is necessary to incorporate some of the advances. After consultation with those who adopt this book, we have decided to continue to use the 68K family to introduce the computer instruction set because this processor still has one of the most sophisticated of all instruction set architectures.

ACKNOWLEDGEMENTS

Few books are entirely the result of one person's unaided efforts and this is no exception. I would like to thank all those who wrote the books about computers on which my own understanding is founded. Some of these writers conveyed the sheer fascination of computer architecture that was to change the direction of my own academic career. It really is amazing how a large number of gates (a circuit element whose operation is so simple as to be trivial) can be arranged in such a way as to perform all the feats we associate computers with today.

I am grateful for all the comments and feedback I've received from my wife, colleagues, students, and reviewers over the years. Their feedback has helped me to improve the text and eliminate some of the errors I'd missed in editing. More importantly, their help and enthusiasm has made the whole project worthwhile.

Although I owe a debt of gratitude to a lot of people, I would like to mention four people who have had a considerable

impact. Alan Knowles of Manchester University read drafts of both the second and third editions with a precision well beyond that of the average reviewer. Paul Lambert, one of my colleagues at The University of Teesside, wrote the 68K cross-assembler and simulator that I use in my teaching. In this edition we have used a Windows-based graphical 68K simulator kindly provided by Charles Kelly.

Dave Barker, one of my former students and an excellent programmer, wrote the logic simulator called Digital Works that accompanies this book. I would particularly like to thank Dave for providing a tool that enables students to construct circuits and test them without having to connect wires together.

One of the major changes to the third edition was the chapter on the ARM processor. I would like to thank Steve Furber of Manchester University (one of the ARM's designers) for encouraging me to use this very interesting device.

CONTENTS

1 Introduction to computer hardware	1	2.5 An introduction to Boolean algebra	56
1.1 What is computer hardware?	1	2.5.1 Axioms and theorems of Boolean algebra	56
1.2 Why do we teach computer hardware?	2	2.5.2 De Morgan's theorem	63
1.2.1 Should computer architecture remain in the CS curriculum?	3	2.5.3 Implementing logic functions in NAND or NOR two logic only	65
1.2.2 Supporting the CS curriculum	4	2.5.4 Karnaugh maps	67
1.3 An overview of the book	5	2.6 Special-purpose logic elements	83
1.4 History of computing	6	2.6.1 The multiplexer	84
1.4.1 Navigation and mathematics	6	2.6.2 The demultiplexer	84
1.4.2 The era of mechanical computers	6	2.7 Tri-state logic	87
1.4.3 Enabling technology—the telegraph	8	2.7.1 Buses	88
1.4.4 The first electromechanical computers	10	2.8 Programmable logic	91
1.4.5 The first mainframes	11	2.8.1 The read-only memory as a logic element	91
1.4.6 The birth of transistors, ICs, and microprocessors	12	2.8.2 Programmable logic families	93
1.4.7 Mass computing and the rise of the Internet	14	2.8.3 Modern programmable logic	94
1.5 The digital computer	15	2.8.4 Testing digital circuits	96
1.5.1 The PC and workstation	15	SUMMARY	98
1.5.2 The computer as a data processor	15	PROBLEMS	98
1.5.3 The computer as a numeric processor	16		
1.5.4 The computer in automatic control	17		
1.6 The stored program computer—an overview	19	3 Sequential logic	101
1.7 The PC—a naming of parts	22	3.1 The RS flip-flop	103
SUMMARY	23	3.1.1 Analyzing a sequential circuit by assuming initial conditions	104
PROBLEMS	23	3.1.2 Characteristic equation of an RS flip-flop	105
		3.1.3 Building an RS flip-flop from NAND gates	106
2 Gates, circuits, and combinational logic	25	3.1.4 Applications of the RS flip-flop	106
2.1 Analog and digital systems	26	3.1.5 The clocked RS flip-flop	108
2.2 Fundamental gates	28	3.2 The D flip-flop	109
2.2.1 The AND gate	28	3.2.1 Practical sequential logic elements	110
2.2.2 The OR gate	30	3.2.2 Using D flip-flops to create a register	110
2.2.3 The NOT gate	31	3.2.3 Using Digital Works to create a register	111
2.2.4 The NAND and NOR gates	31	3.2.4 A typical register chip	112
2.2.5 Positive, negative, and mixed logic	32	3.3 Clocked flip-flops	113
2.3 Applications of gates	34	3.3.1 Pipelining	114
2.4 Introduction to Digital Works	40	3.3.2 Ways of clocking flip-flops	115
2.4.1 Creating a circuit	41	3.3.3 Edge-triggered flip-flops	116
2.4.2 Running a simulation	45	3.3.4 The master-slave flip-flop	117
2.4.3 The clock and sequence generator	48	3.3.5 Bus arbitration—an example	118
2.4.4 Using Digital Works to create embedded circuits	50	3.4 The JK flip-flop	120
2.4.5 Using a macro	52	3.5 Summary of flip-flop types	121

3.6 Applications of sequential elements	122	5 The instruction set architecture	203
3.6.1 Shift register	122	5.1 What is an instruction set architecture?	204
3.6.2 Asynchronous counters	128	5.2 Introduction to the CPU	206
3.6.3 Synchronous counters	132	5.2.1 Memory and registers	207
3.7 An introduction to state machines	134	5.2.2 Register transfer language	208
3.7.1 Example of a state machine	136	5.2.3 Structure of the CPU	209
3.7.2 Constructing a circuit to implement the state table	138	5.3 The 68K family	210
SUMMARY	139	5.3.1 The instruction	210
PROBLEMS	140	5.3.2 Overview of addressing modes	215
<hr/>		5.4 Overview of the 68K's instructions	217
4 Computer arithmetic	145	5.4.1 Status flags	217
4.1 Bits, bytes, words, and characters	146	5.4.2 Data movement instructions	218
4.2 Number bases	148	5.4.3 Arithmetic instructions	218
4.3 Number base conversion	150	5.4.4 Compare instructions	220
4.3.1 Conversion of integers	150	5.4.5 Logical instructions	220
4.3.2 Conversion of fractions	152	5.4.6 Bit instructions	221
4.4 Special-purpose codes	153	5.4.7 Shift instructions	221
4.4.1 BCD codes	153	5.4.8 Branch instructions	223
4.4.2 Unweighted codes	154	SUMMARY	226
4.5 Error-detecting codes	156	PROBLEMS	226
4.5.1 Parity EDCs	158	<hr/>	
4.5.2 Error-correcting codes	158	6 Assembly language programming	228
4.5.3 Hamming codes	160	6.1 Structure of a 68K assembly language program	228
4.5.4 Hadamard codes	161	6.1.1 Assembler directives	229
4.6 Data-compressing codes	163	6.1.2 Using the cross-assembler	232
4.6.1 Huffman codes	164	6.2 The 68K's registers	234
4.6.2 Quadrees	167	6.2.1 Data registers	235
4.7 Binary arithmetic	169	6.2.2 Address registers	236
4.7.1 The half adder	170	6.3 Features of the 68K's instruction set	237
4.7.2 The full adder	171	6.3.1 Data movement instructions	237
4.7.3 The addition of words	173	6.3.2 Using arithmetic operations	241
4.8 Signed numbers	175	6.3.3 Using shift and logical operations	244
4.8.1 Sign and magnitude representation	176	6.3.4 Using conditional branches	244
4.8.2 Complementary arithmetic	176	6.4 Addressing modes	249
4.8.3 Two's complement representation	177	6.4.1 Immediate addressing	249
4.8.4 One's complement representation	180	6.4.2 Address register indirect addressing	250
4.9 Floating point numbers	181	6.4.3 Relative addressing	259
4.9.1 Representation of floating point numbers	182	6.5 The stack	262
4.9.2 Normalization of floating point numbers	183	6.5.1 The 68K stack	263
4.9.3 Floating point arithmetic	186	6.5.2 The stack and subroutines	266
4.9.4 Examples of floating point calculations	188	6.5.3 Subroutines, the stack, and parameter passing	271
4.10 Multiplication and division	189	6.6 Examples of 68K programs	280
4.10.1 Multiplication	189	6.6.1 A circular buffer	282
4.10.2 Division	194	SUMMARY	287
SUMMARY	198	PROBLEMS	287
PROBLEMS	198		

7 Structure of the CPU	293		
7.1 The CPU	294		
7.1.1 The address path	294		
7.1.2 Reading the instruction	295		
7.1.3 The CPU's data paths	296		
7.1.4 Executing conditional instructions	298		
7.1.5 Dealing with literal operands	300		
7.2 Simulating a CPU	300		
7.2.1 CPU with an 8-bit instruction	301		
7.2.2 CPU with a 16-bit instruction	304		
7.3 The random logic control unit	308		
7.3.1 Implementing a primitive CPU	308		
7.3.2 From op-code to operation	312		
7.4 Microprogrammed control units	315		
7.4.1 The microprogram	316		
7.4.2 Microinstruction sequence control	319		
7.4.3 User-microprogrammed processors	320		
SUMMARY	322		
PROBLEMS	322		
8 Accelerating performance	325		
8.1 Measuring performance	326		
8.1.1 Comparing computers	326		
8.2 The RISC revolution	327		
8.2.1 Instruction usage	328		
8.2.2 Characteristics of RISC architectures	329		
8.3 RISC architecture and pipelining	335		
8.3.1 Pipeline hazards	336		
8.3.2 Data dependency	338		
8.3.3 Reducing the branch penalty	339		
8.3.4 Implementing pipelining	341		
8.4 Cache memory	344		
8.4.1 Effect of cache memory on computer performance	345		
8.4.2 Cache organization	346		
8.4.3 Considerations in cache design	350		
8.5 Multiprocessor systems	350		
8.5.1 Topics in Multiprocessor Systems	352		
8.5.2 Multiprocessor organization	353		
8.5.3 MIMD architectures	356		
SUMMARY	362		
PROBLEMS	362		
9 Processor architectures	365		
9.1 Instruction set architectures and their resources	365		
9.1.1 Register sets	365		
9.1.2 Instruction formats	366		
9.1.3 Instruction types	366		
9.1.4 Addressing modes	367		
9.1.5 On-chip peripherals	367		
9.2 The microcontroller	367		
9.2.1 The M68HC12	368		
9.3 The ARM—an elegant RISC processor	375		
9.3.1 ARM's registers	375		
9.3.2 ARM instructions	377		
9.3.3 ARM branch instructions	380		
9.3.4 Immediate operands	381		
9.3.5 Sequence control	381		
9.3.6 Data movement and memory reference instructions	382		
9.3.7 Using the ARM	385		
SUMMARY	397		
PROBLEMS	398		
10 Buses and input/output mechanisms	399		
10.1 The bus	400		
10.1.1 Bus architecture	400		
10.1.2 Key bus concepts	400		
10.1.3 The PC bus	404		
10.1.4 The IEEE 488 bus	407		
10.1.5 The USB serial bus	411		
10.2 I/O fundamentals	412		
10.2.1 Programmed I/O	413		
10.2.2 Interrupt-driven I/O	415		
10.3 Direct memory access	422		
10.4 Parallel and serial interfaces	423		
10.4.1 The parallel interface	424		
10.4.2 The serial interface	428		
SUMMARY	433		
PROBLEMS	433		
11 Computer Peripherals	435		
11.1 Simple input devices	436		
11.1.1 The keyboard	436		
11.1.2 Pointing devices	440		
11.2 CRT, LED, and plasma displays	444		
11.2.1 Raster-scan displays	445		
11.2.2 Generating a display	445		
11.2.3 Liquid crystal and plasma displays	447		
11.2.4 Drawing lines	450		
11.3 The printer	452		
11.3.1 Printing a character	453		
11.3.2 The Inkjet printer	453		
11.3.3 The laser printer	455		

xiv Contents

11.4 Color displays and printers	457	12.7.3 RAID systems	531
11.4.1 Theory of color	457	12.7.4 The floppy disk drive	532
11.4.2 Color CRTs	458	12.7.5 Organization of data on disks	533
11.4.3 Color printers	460	12.8 Optical memory technology	536
11.5 Other peripherals	461	12.8.1 Storing and reading information	537
11.5.1 Measuring position and movement	461	12.8.2 Writable CDs	540
11.5.2 Measuring temperature	463	SUMMARY	543
11.5.3 Measuring light	464	PROBLEMS	543
11.5.4 Measuring pressure	464		
11.5.5 Rotation sensors	464	<hr/>	
11.5.6 Biosensors	465	13 The operating system	547
<hr/>			
11.6 The analog interface	466	13.1 The operating system	547
11.6.1 Analog signals	466	13.1.1 Types of operating system	548
11.6.2 Signal acquisition	467	13.2 Multitasking	550
11.6.3 Digital-to-analog conversion	473	13.2.1 What is a process?	551
11.6.4 Analog-to-digital conversion	477	13.2.2 Switching processes	551
11.7 Introduction to digital signal processing	486	13.3 Operating system support from the CPU	554
11.7.1 Control systems	486	13.3.1 Switching states	555
11.7.2 Digital signal processing	488	13.3.2 The 68K's two Stacks	556
SUMMARY	491	13.4 Memory management	561
PROBLEMS	492	13.4.1 Virtual memory	563
<hr/>		13.4.2 Virtual memory and the 68K family	565
12 Computer memory	493	SUMMARY	568
		PROBLEMS	568
<hr/>			
12.1 Memory hierarchy	493	<hr/>	
12.2 What is memory?	496	14 Computer communications	569
12.3 Memory technology	496		
12.3.1 Structure modification	496	14.1 Background	570
12.3.2 Delay lines	496	14.1.1 Local area networks	571
12.3.3 Feedback	496	14.1.2 LAN network topology	572
12.3.4 Charge storage	497	14.1.3 History of computer communications	574
12.3.5 Magnetism	498	14.2 Protocols and computer communications	576
12.3.6 Optical	498	14.2.1 Standards bodies	578
12.4 Semiconductor memory	498	14.2.2 Open systems and standards	578
12.4.1 Static semiconductor memory	498	14.3 The physical layer	584
12.4.2 Accessing memory—timing diagrams	499	14.3.1 Serial data transmission	584
12.4.3 Dynamic memory	501	14.4 The PSTN	587
12.4.4 Read-only semiconductor memory devices	505	14.4.1 Channel characteristics	587
12.5 Interfacing memory to a CPU	506	14.4.2 Modulation and data transmission	588
12.5.1 Memory organization	507	14.4.3 High-speed transmission over the PSTN	591
12.5.2 Address decoders	508	14.5 Copper cable	592
12.6 Secondary storage	515	14.5.1 Ethernet	593
12.6.1 Magnetic surface recording	515	14.6 Fiber optic links	595
12.6.2 Data encoding techniques	521	14.7 Wireless links	596
12.7 Disk drive principles	524	14.7.1 Spread spectrum technology	598
12.7.1 Disk drive operational parameters	527		
12.7.2 High-performance drives	529		

14.8 The data link layer	599	SUMMARY	609
14.8.1 Bit-oriented protocols	599	PROBLEMS	610
14.8.2 The Ethernet data link layer	603		
14.9 Routing techniques	604	Appendix: The 68000 instruction set	611
14.9.1 Centralized routing	607	Bibliography	641
14.9.2 Distributed routing	607	Index	643
14.9.3 IP (Internet protocol)	607	Contents and installation instructions for the CD-Rom	653