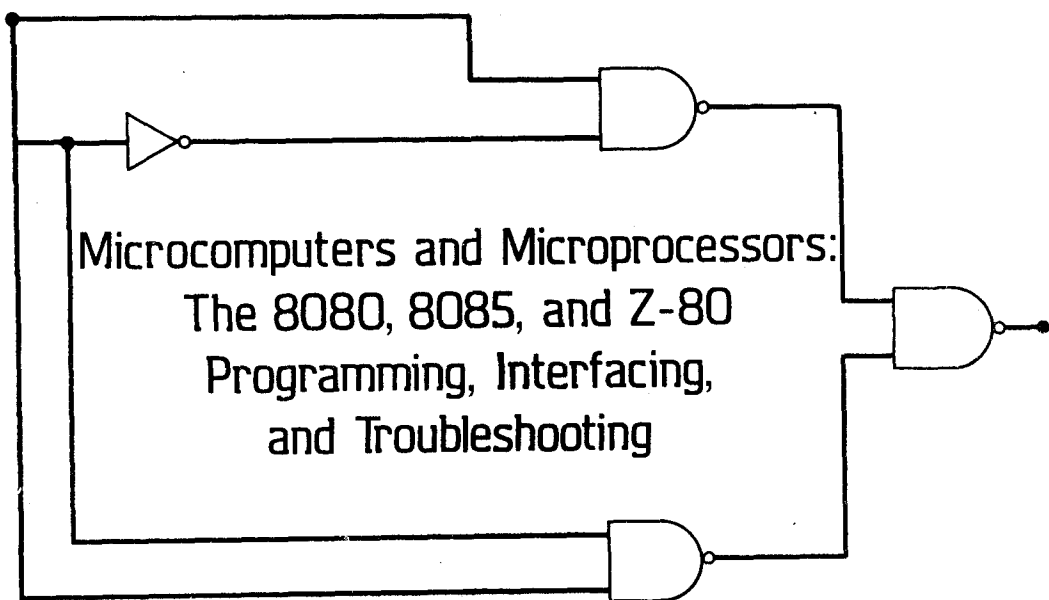
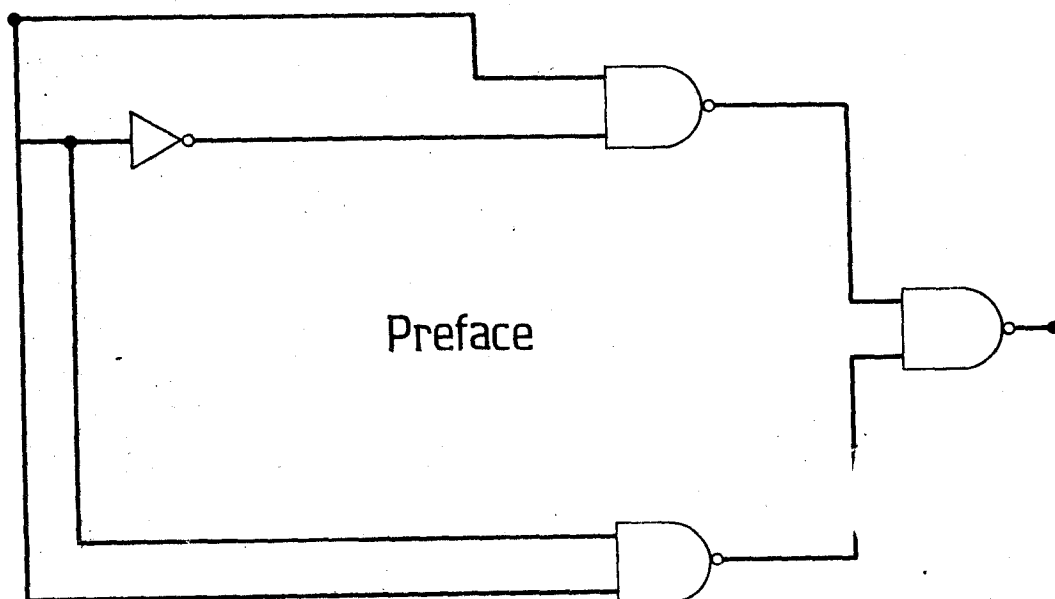


John Uffenbeck



John Uffenbeck

Prentice-Hall, Inc., Englewood Cliffs, N.J. 07632



In writing this book I had two main goals in mind:

1. Provide a practical microcomputer text with an emphasis on the programmable peripheral controller chips now available to support the microprocessor.
2. Use the Intel 8080, 8085, and the Zilog Z-80 microprocessors as realistic "vehicles" to demonstrate these concepts.

You will find *Microcomputers and Microprocessors* an easy book to learn or teach from. It is written in a casual, yet easily understood style that belies its thorough coverage. There are more than 100 examples and 400 homework problems. Problems marked with a (*) are more advanced, requiring additional time and study. Many of these problems are also suitable as laboratory exercises. A chapter summary ensures that you do not miss any key concepts presented in that chapter.

I do not assume any previous computer knowledge and therefore begin Chap. 1 with an introduction to the stored program computer. By the end of this chapter the three-bus system architecture has been introduced and machine cycle timing diagrams used to illustrate memory and I/O read and write cycles.

Particular emphasis is placed on visualizing the microprocessor as part of a *CPU module* that is, in turn, interfaced by the memory and I/O. Chapters 2 and 4 develop this module, paying close attention to bus buffering concepts such as noise immunity, dc loading, and reflections.

The CPU module concept is stressed throughout the book, even to including the 16-bit 8086 microprocessor in Chap. 12. In this way the 16-bit processor can be seen as an extension of the 8-bit processors studied in previous chapters.

Like learning a foreign language, microprocessor programming is best learned *in context*. This is accomplished in Chap. 3 with 14 detailed programs each chosen to illustrate new instructions and addressing modes while emphasizing practical applications of the microprocessor. For example, a serial communications test program is given that illustrates bit-testing techniques and is also an effective test of a serial communications link. Also included in Chap. 3 is a brief introduction to CP/M, an industry-wide operating system for 8080/85 and Z-80 microcomputer systems.

The software is presented early in the book so that readers with access to hardware supporting one of the three processors can begin to get immediate "hands-on" experience. In addition, the later chapters make extensive use of software to control the hardware designed in these chapters. All told, there are more than 50 commented programs included in the text.

Chapter 5 shows how to interface RAM and ROM memory to the CPU modules developed earlier. Timing requirements imposed by the microprocessor are defined and the memory designs analyzed to verify that these constraints have been met.

Chapter 6 concludes "construction" of the microcomputer with the addition of the I/O module. The microprocessor IN and OUT instructions are used to identify the I/O-mapped and memory-mapped hardware requirements. The three common methods of controlling I/O devices—polling, interrupts, and DMA—are contrasted and response-time and transfer rates are calculated for each.

In some cases the three microprocessors are sufficiently different to warrant separate chapters. This is the case in Chaps. 7 and 8. Chapter 7 covers the common peripheral controller chips designed in support of the 8080 and 8085. Chapter 8 discusses similar chips supporting the Z-80. In total, 11 devices are discussed in detail, ranging from the 8755A EPROM with I/O designed for the 8085 to the WD2793 floppy disk formatter/controller designed to simplify the task of interfacing a floppy disk drive.

Much of this book deals with microcomputer concepts that are not unique to the 8080, 8085, or Z-80 microprocessors. This is especially true in Chaps. 9 through 11. In these chapters material is included on serial communications, telecommunications, analog interfacing, and floppy disk drive specifications and interfacing.

As microcomputer hardware becomes more sophisticated, so do the troubleshooting techniques required to test that equipment. Chapter 11 introduces the logic probe, signature analyzer, and logic analyzer and discusses the application of each of these tools.

Chapter 12 is a "mini" text in itself, retracing Chaps. 1 through 6 with the 16-bit 8086 microprocessor. After developing min and max mode CPU modules, memory and I/O modules are interfaced and contrasted with the similar circuits designed for the 8-bit processors. The emphasis is on comparing and contrasting 16-bit microprocessor concepts to the 8-bit concepts already learned.

The 8086 instruction set is also surveyed in the form of several tables that can be closely studied or skimmed as the reader desires.

In writing this book I have assumed a background in digital electronics through the gates and flip-flops level. A brief review of hexadecimal arithmetic is presented in Chap. 1, as this number system is used extensively throughout the book. Chapter

4 covers loading rules for the TTL logic family for readers unfamiliar with this topic.

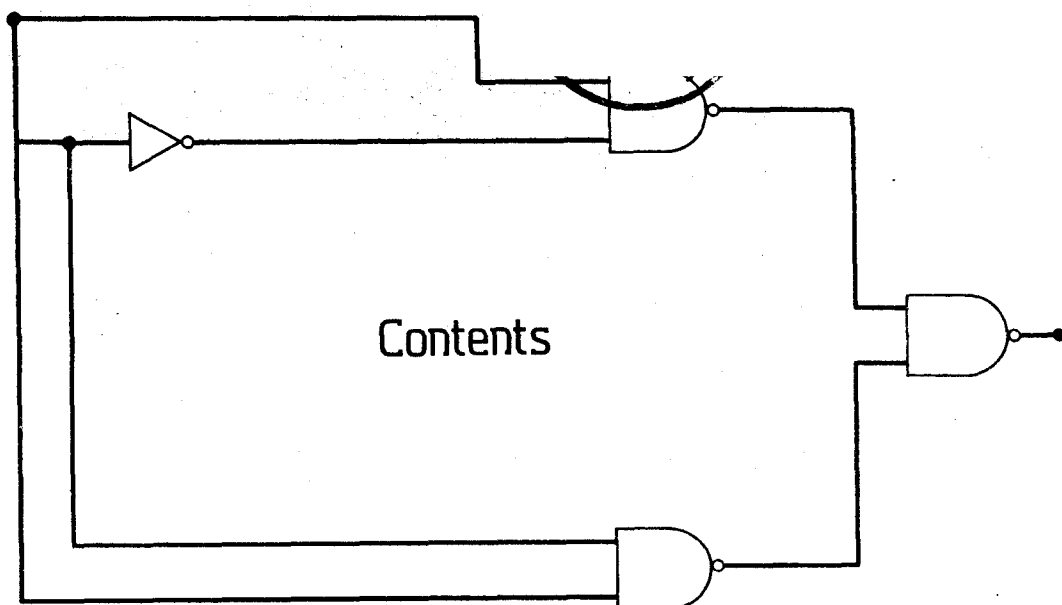
Recognizing that we must all learn to read the literature distributed by the semiconductor manufacturers, portions of over 40 data sheets are distributed throughout the text as appropriate. An appendix is also provided with handy references to hexadecimal instruction set summaries for the 8080, 8085, and Z-80.

Microcomputers and Microprocessors should be of interest to anyone using one of the three processors discussed. However, with the thorough coverage of memory technologies, serial communications, magnetic disk concepts, analog interfacing, and microcomputer troubleshooting, the book is more a microcomputer text than it is a book specifically about the 8080, 8085, or Z-80.

As a textbook, it can be used in a one- or two-semester course following the digital electronics sequence. Material that is unique to one of the three processors is usually highlighted so that readers interested in only one processor can skip inappropriate material. However, you may find yourself learning all three processors for the price of one!

A number of people and companies assisted me in the writing of this book and I would like to thank them all. I would particularly like to thank the literature departments of Intel Corporation and Zilog, Inc., for permission to reproduce the large number of data sheets describing their products. Finally, I would like to thank all of the students I have had over the years, particularly at Hartnell College and the University of Wisconsin-Platteville, where my "real" learning took place. It is to these students that I dedicate this book.

John Uffenbeck



PREFACE

xiii

1 INTRODUCTION TO THE MICROPROCESSOR

1

- 1.1 Digital Computers: Some Basics 2
 - The stored program computer* 2
 - Fetch and execute* 2
 - The three-bus architecture* 4
 - Computer programming* 4
- 1.2 Computer Codes 5
 - Bits, bytes, and words* 6
 - Binary, decimal, and hexadecimal numbers* 6
 - Codes* 10
- 1.3 Computer Languages 11
 - Machine and assembly language programming* 12
 - High-level languages* 13
- 1.4 Implementing the Three-Bus Architecture in Hardware 15
 - Digital signals* 15
 - Defining the three buses* 18
- 1.5 The CPU as a Complex Timer 18
 - Machine cycle timing diagrams* 20
 - Instruction timing* 22
 - Processor timing* 22
- Chapter Summary 25
- Questions and Problems 26

2 INTRODUCING THE 8080, 8085, AND Z-80 MICROPROCESSORS

29

- 2.1 Constructing the CPU Module 29
 - The CPU module* 30
- 2.2 CPU Modules for the 8080, 8085, and Z-80 Microprocessors 32
 - The 8080 CPU module* 32
 - The 8085 CPU module* 38
 - The Z-80 CPU module* 44
- 2.3 Programming Models for the 8080, 8085, and Z-80 Microprocessors 46
 - A programming model for the 8080* 48
 - The 8080 flag register* 49
 - A programming model for the 8085* 51
 - A programming model for the Z-80* 51
 - The Z-80 flag register* 52
- 2.4 Introducing the Instruction Sets 53
 - Instruction types* 53
- 2.5 Addressing Modes 62
- 2.6 Putting It All Together: A Programming Example 64
- Chapter Summary 66
- Questions and Problems 67

3 PROGRAMMING THE MICROPROCESSOR

71

- 3.1 Microprocessor Programming Examples 72
 - Program 1: 8080/85 8-bit addition* 72
 - Program 2: Z-80 8-bit addition* 74
 - Program 3: 32-bit binary addition* 76
 - Program 4: 32-bit decimal addition* 79
 - Program 5: 8-bit multiplication* 82
 - Program 6: BCD-to-binary conversion* 86
 - Program 7: filling a block of memory* 91
 - Program 8: square-wave generator* 94
 - Program 9: serial communications test program* 95
 - Program 10: hex dump* 97
 - Program 11: 1-bit I/O port* 102
 - Program 12: frequency counter* 104
 - Program 13: the game of nim* 107
 - Program 14: computer music* 113
- 3.2 Operating Systems 119
 - Some common operating systems* 119
 - Features of CP/M* 120
 - A sample session with CP/M* 122
 - Linking programs to CP/M* 126
- Chapter Summary 130
- Questions and Problems 131

4	BUILDING THE MICROCOMPUTER, PART 1: THE BUSES	134
4.1	Generating the System Clock 134	
	<i>The 8080 clock</i> 138	
	<i>The 8085 clock</i> 140	
	<i>The Z-80 clock</i> 140	
4.2	Resetting the Microprocessor 141	
	<i>Starting up a "new" computer</i> 141	
	<i>Reset circuits for the 8080, 8085, and Z-80</i> 141	
4.3	Electrical Characteristics of a Bus 142	
	<i>Noise immunity</i> 143	
	<i>Bus loading</i> 144	
	<i>Reflections</i> 146	
4.4	Bus Buffering Techniques 148	
	<i>Type 1 bus</i> 148	
	<i>Tri-state buffers with hysteresis</i> 149	
	<i>Type 2 bus</i> 153	
	<i>Type 3 bus</i> 155	
4.5	CPU Modules for the 8080, 8085, and Z-80 160	
	<i>The 8080 CPU module</i> 160	
	<i>The 8085 CPU module</i> 160	
	<i>The Z-80 CPU module</i> 163	
	<i>Summary</i> 163	
4.6	Single-Stepping the Microprocessor 165	
	<i>Single-stepping the 8080</i> 166	
	<i>Single-stepping the 8085</i> 167	
	<i>Single-stepping the Z-80</i> 167	
4.7	A Power-On-Jump Circuit for the Z-80 170	
	Chapter Summary 172	
	Questions and Problems 173	
5	BUILDING THE MICROCOMPUTER, PART 2: ADDING MEMORY	175
5.1	Memory Hierarchies 176	
5.2	The Microprocessor Defines the Memory Timing 178	
	<i>Memory-read-cycle timing</i> 178	
	<i>Memory-write-cycle timing</i> 179	
	<i>Memory interfacing requirements</i> 182	
	<i>Interfacing slow memory</i> 182	
5.3	Choosing Memory 184	
	<i>ROM applications</i> 184	
	<i>RAM applications</i> 185	
	<i>The memory map</i> 185	
5.4	RAM and ROM Technologies 186	
	<i>Mask-programmable ROMs</i> 186	
	<i>Field-programmable ROMs</i> 187	

Static and dynamic RAMs	196
RAM organization	199
The universal site	200
5.5 Interfacing Static RAM and ROM to the Microprocessor	201
Interfacing the 2764 8K-byte EPROM	202
Interfacing the 2167 16K static RAM	207
Interfacing a RAM/ROM module	211
5.6 Interfacing Dynamic RAM to the Microprocessor	215
Timing diagrams for dynamic RAM	216
Refresh	219
The 8203 DRAM controller	221
The Z-80 as a refresh controller	224
5.7 Conclusion	224
Chapter Summary	225
Questions and Problems	226
 6 BUILDING THE MICROCOMPUTER, PART 3: INPUT/OUTPUT	 232
6.1 Parallel I/O: Interfacing to a Type 3 Bus	233
I/O machine cycles and timing	233
Designing an 8-bit input port	235
Designing an 8-bit output port	238
Applications for the device select pulse	242
6.2 Memory-Mapped I/O	243
Designing an 8-bit memory-mapped input port	243
Designing a digital lock	245
6.3 Handshaking Logic	249
Busy, ready, and acknowledge flags	249
6.4 Programmed I/O	251
Polling	251
Data transfer rate	254
Priorities	254
6.5 Interrupt-Driven I/O	257
Generating an interrupt	258
Maskable and nonmaskable interrupts	259
Branching to the interrupt service routine	261
Response time and transfer rate	265
Multiple interrupts: the priority problem	268
Summary points for the 8080, 8085, and Z-80	272
6.6 Direct Memory Access	273
Chapter Summary	277
Questions and Problems	278
 7 SPECIAL-PURPOSE SUPPORT DEVICES: THE 8080/85 FAMILY	 284
7.1 The 8755A 16K EPROM with I/O	285
Interfacing the 8755A ROM	286

	<i>Interfacing the 8755A I/O ports</i>	287
	<i>Three-chip 8085 microcomputer system</i>	290
7.2	The 8255A Programmable Peripheral Interface	292
	<i>Interfacing the 8255A</i>	293
	<i>Mode 0: basic I/O</i>	294
	<i>The bit set/reset mode</i>	297
	<i>Electrical characteristics of the ports</i>	299
	<i>Mode 1: strobed I/O</i>	300
	<i>Mode 2: strobed bidirectional I/O</i>	307
7.3	The 8254 Programmable Interval Timer	312
	<i>Interfacing the 8254</i>	312
	<i>Programming the 8254</i>	314
	<i>Mode definitions</i>	317
	<i>A design example</i>	318
	<i>8254 electrical characteristics</i>	320
7.4	The 8259A Programmable Interrupt Controller	320
	<i>Interfacing the 8259A</i>	321
	<i>Arbitration modes</i>	322
	<i>Programming the 8259A</i>	326
	<i>Operation control words</i>	329
7.5	The 8237 Programmable DMA Controller	332
	<i>Interfacing the 8237A</i>	333
	<i>Response time and transfer rate</i>	337
	<i>Programming the 8237</i>	340
	<i>A design example: interfacing a floppy disk drive</i>	346
7.6	Peripheral Controller Bus Buffering Techniques	351
	Chapter Summary	353
	Questions and Problems	354

8 SPECIAL SUPPORT DEVICES: THE Z-80 FAMILY 358

8.1	The Z8420 Parallel Input/Output Controller	359
	<i>Interfacing the Z-80 PIO</i>	360
	<i>Programming the Z-80 PIO</i>	362
	<i>Mode 0: output port with handshake</i>	365
	<i>Mode 1: input port with handshake</i>	367
	<i>Using the Z-80 PIO to interface a parallel printer</i>	367
	<i>Mode 2: bidirectional I/O with handshake</i>	370
	<i>Mode 3: bit-defined I/O</i>	372
	<i>A design example: mode 3 control of a multiplexed LED display</i>	372
	<i>Electrical characteristics of the ports</i>	377
8.2	The Z8430 Counter/Timer Circuit	377
	<i>Interfacing the Z-80 CTC</i>	378
	<i>Programming the Z-80 CTC: counter mode</i>	379
	<i>Programming the Z-80 CTC: timer mode</i>	383
	<i>Electrical characteristics</i>	388

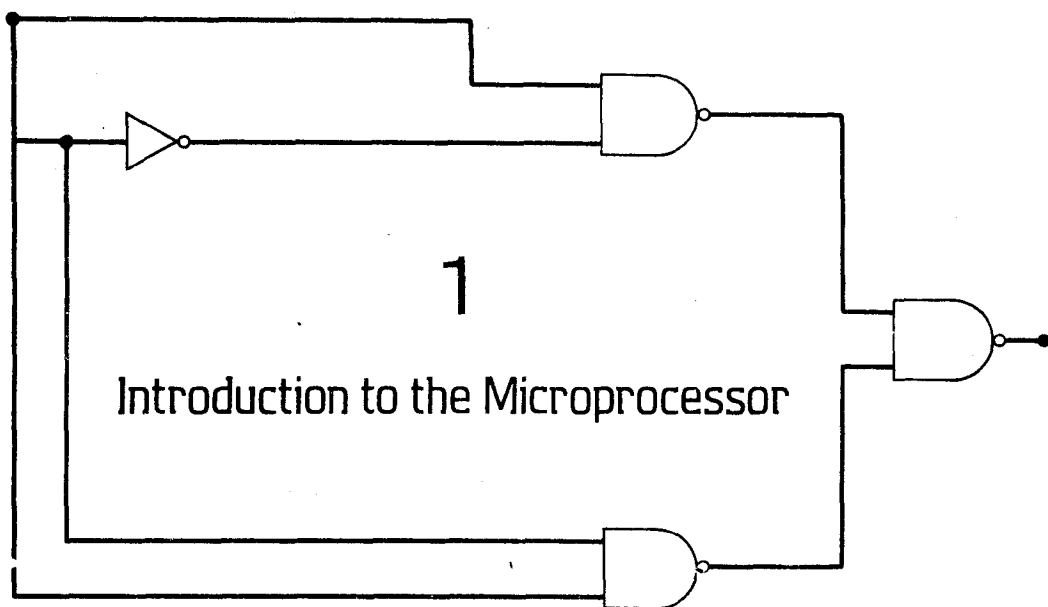
8.3	The Z8410 Direct Memory Access Controller	389
	<i>Interfacing the Z-80 DMA</i>	389
	<i>Typical DMA transfer</i>	390
	<i>Response time and transfer rate</i>	393
	<i>Read/write registers</i>	395
	<i>Programming the Z-80 DMA</i>	398
8.4	Peripheral Controller Bus Buffering Techniques	400
	Chapter Summary	400
	Questions and Problems	401

9 SERIAL I/O TECHNIQUES 404

9.1	Asynchronous Serial Communications	405
	<i>Start bits, stop bits, and the baud rate</i>	405
	<i>Generating and recovering asynchronous serial data</i>	406
	<i>Standard asynchronous serial communications protocols</i>	409
	<i>The UART</i>	410
9.2	Synchronous Serial Communications	414
	<i>Bisync protocol</i>	414
	<i>Serial data link control</i>	415
9.3	Error Detection and Correction	416
	<i>Parity</i>	416
	<i>Checksums</i>	417
	<i>Cyclic redundancy checks</i>	418
	<i>The Hamming code</i>	419
9.4	The Intel 8251A USART	425
	<i>Interfacing the 8251A</i>	425
	<i>Programming the 8251A: asynchronous mode</i>	429
	<i>Programming the 8251A: synchronous mode</i>	430
9.5	The Zilog Z-80 SIO and Z-80 DART	433
	<i>Comparing the SIO and the DART</i>	433
	<i>Interfacing the Z-80 SIO</i>	435
	<i>Programming the Z-80 SIO: asynchronous mode</i>	437
	<i>Controlling the Z-80 SIO in the asynchronous mode</i>	441
	<i>Using the Z-80 SIO in the synchronous mode</i>	446
9.6	Remote Control Applications for Asynchronous Serial Data	448
9.7	Serial Data Interface Standards	449
	<i>The EIA RS-232C standard</i>	450
	<i>The RS-422A and RS-423A standards</i>	459
9.8	Telecommunications	460
	<i>The basics</i>	460
	<i>Interfacing a 300-bps modem</i>	463
	<i>High-speed modems</i>	467
	Chapter Summary	469
	Questions and Problems	470

10	SECONDARY STORAGE TECHNIQUES: FLOPPY AND RIGID DISK TECHNOLOGIES	475
10.1	The Media 476 <i>Tracks, sectors, and storage capacity</i> 477 <i>Hard and soft sectoring</i> 479	
10.2	The Disk Drive 480 <i>Floppy-disk drives</i> 480 <i>Winchester disk drives</i> 489	
10.3	Encoding Techniques 492 <i>Single-density encoding</i> 492 <i>Double-density encoding</i> 493 <i>Peak-shift distortion</i> 495	
10.4	Disk Formatting 498 <i>The IBM 3740 single-density format</i> 499 <i>The IBM system 34 double-density format</i> 501	
10.5	The Western Digital WD2793 Floppy-Disk Controller/ Formatter 503 <i>Floppy-disk-drive interface signals</i> 503 <i>Interfacing the WD2793 floppy-disk controller/ formatter</i> 505 <i>WD2793 register organization and command words</i> 508 <i>Programming the WD2793</i> 512	
10.6	The Disk Operating System 519	
	Chapter Summary 521	
	Questions and Problems 522	
11	MICROCOMPUTER CONTROL APPLICATIONS AND TROUBLESHOOTING TECHNIQUES	524
11.1	Detecting the Presence of an Analog Signal: The Comparator 525	
11.2	ON/OFF Control of Analog Peripherals 528 <i>DC control</i> 528 <i>AC control</i> 533	
11.3	Interfacing a Digital-to-Analog Converter 537 <i>The digital-to-analog conversion process</i> 537 <i>Interfacing the MC1408 DAC</i> 539 <i>Interfacing the DAC1200</i> 544	
11.4	Interfacing an Analog-to-Digital Converter 546 <i>The analog-to-digital conversion process</i> 547 <i>Interfacing the ADC0809 eight-channel ADC</i> 553	
11.5	Troubleshooting Techniques 559 <i>Hardware troubleshooting tools</i> 560 <i>Summary</i> 568	
	Chapter Summary 569	
	Questions and Problems 569	

12	INTRODUCTION TO THE 8086 16-BIT MICROPROCESSOR	573
12.1	8086 Hardware Details and Basic System Timing	574
	<i>The queue</i>	576
	<i>The min and max mode</i>	576
	<i>Memory organization</i>	576
	<i>Basic system timing</i>	579
	<i>Special support chips</i>	582
12.2	Min and Max Mode CPU Modules for the 8086	583
	<i>The min mode</i>	584
	<i>The max mode</i>	585
12.3	A Programming Model for the 8086	587
	<i>Internal register array</i>	587
	<i>Segment registers</i>	589
	<i>Addressing modes</i>	592
12.4	Programming the 8086	594
	<i>Data transfer group</i>	594
	<i>Arithmetic group</i>	595
	<i>Bit manipulation group</i>	597
	<i>Transfer group</i>	597
	<i>Interrupts</i>	600
	<i>String group</i>	602
	<i>Processor control group</i>	603
	<i>An example: two 8086 block fill programs</i>	603
12.5	8086 Memory and I/O Interfacing	605
	<i>Interfacing a 16K-word memory</i>	605
	<i>Interfacing the 8255A PPI</i>	606
12.6	The 8088 Microprocessor	610
	Chapter Summary	611
	Questions and Problems	612
	ANSWERS TO SELECTED PROBLEMS	615
	APPENDIX	
	Appendix A.1	620
	Appendix A.2	635
	Appendix A.3	636
	Appendix B.1	637
	Appendix C.1	644
	Appendix C.2	647
	Appendix C.3	650
	Appendix D.1	656
	Appendix D.2	659
	Appendix D.3	661
	INDEX	665



What is a *microprocessor*? Some people might answer “a very small computer,” others a “desktop or personal computer,” still others might reply “a computer on a chip.” The word “microprocessor” was coined in the semiconductor industry by Intel Corporation. They used the term to describe a newly designed 4-bit calculator-like integrated circuit.

Today we think of the microprocessor as the silicon chip around which a *microcomputer* is built. Thus we have the IBM Personal Computer, based on the Intel 8088 microprocessor; the Apple IIe, based on the Rockwell 6502; the Radio Shack TRS-80, based on the Zilog Z-80*; and so on.

Some manufacturers have found it advantageous to use several microprocessors in one computer. One might be used to control the keyboard, a second to handle input/output operations, a third to control the mass storage devices (disk drives), and of course, a fourth as the main system processor. This technique is referred to as *distributed processing*.

Some microprocessor chips claim to be single-chip microcomputers. The Zilog Z-8 contains the central processing unit (CPU), a preprogrammed memory containing the operating system software, scratchpad memory for storing temporary results, logic for communicating with a computer terminal, and three parallel input/output ports for hardware control applications.

Despite the advances in semiconductor technology and microprocessors, the basic architecture of the digital computer has remained unchanged for the last 35 years. This is the so-called *von Neumann* model of the stored program computer.

* All figures and tables by Zilog were “Reproduced by permission © 1977, 1978, 1981 Zilog, Inc. This material shall not be reproduced without the written consent of Zilog, Inc.” “Zilog Z-80® is a trademark of Zilog, Inc., with whom Prentice-Hall is not associated.”

In this chapter we study the basic concepts of this model as it applies to the microprocessor.

We will also begin to take a look at the subject of microcomputer programming—perhaps from a point of view you have not taken before. This is to consider the effects each computer instruction has on the electrical lines or “buses” of the microprocessor chip itself.

The chapter concludes with an introduction to instruction timing diagrams. Armed with this information, we can predict exactly how long a given microcomputer program will take to execute (not too long!).

1.1 DIGITAL COMPUTERS: SOME BASICS

In the early 1960s the United States was caught up in a wave of new technology. The transistor, developed by a trio of scientists at Bell Laboratories shortly after the end of World War II, had finally begun to displace its rival, the vacuum tube. Suddenly everything from car radios to electric mixers carried the label *solid-state*.

New electronics companies seemed to spring up overnight, including a company based in Dallas, Texas, called Texas Instruments. TI (as it has become known) had just announced a new solid-state component called the *integrated circuit* (IC). Little did we know then that the electronics version of the industrial revolution was about to begin.

The Stored Program Computer. Figure 1.1 is a block diagram of a typical digital computer. The *central processing unit* or CPU, shown on the far left, is often likened to the human brain because it is here that all decisions are made and the system timing generated. The arithmetic logic unit or ALU is contained within the CPU and all the mathematical operations are performed there. The results of these calculations are left in a special register in the ALU called the *accumulator*.

The memory unit shown in Fig. 1.1 is used to store the specific sequence of commands that will be used to instruct the CPU to perform some task. These instructions are called the computer *program* (hence the name *stored program* computer).

Finally, no useful task can be performed by the computer without the input/output devices—the I/O in “computerese.” It is with the keyboard that we input the instructions or commands about the task to be accomplished. The results are then viewed on the printer or CRT (cathode ray tube) screen.

Studying the memory unit more closely, note that each cell in the memory has its own unique identifying number or *address* and that the total capacity of this particular memory unit is 25 cells.

If we were to examine the contents of these memory cells, we would see a strange collection of numbers having no particular meaning to us. Yet to the CPU these numbers would represent a concise set of commands instructing it to carry out some sequence of operations. These numbers represent the operation codes (*op-codes*) for the various instructions in the CPU’s instruction set.

Fetch and Execute. Continuing to refer to Fig. 1.1, note that the CPU has been designed to follow repeatedly four simple steps.

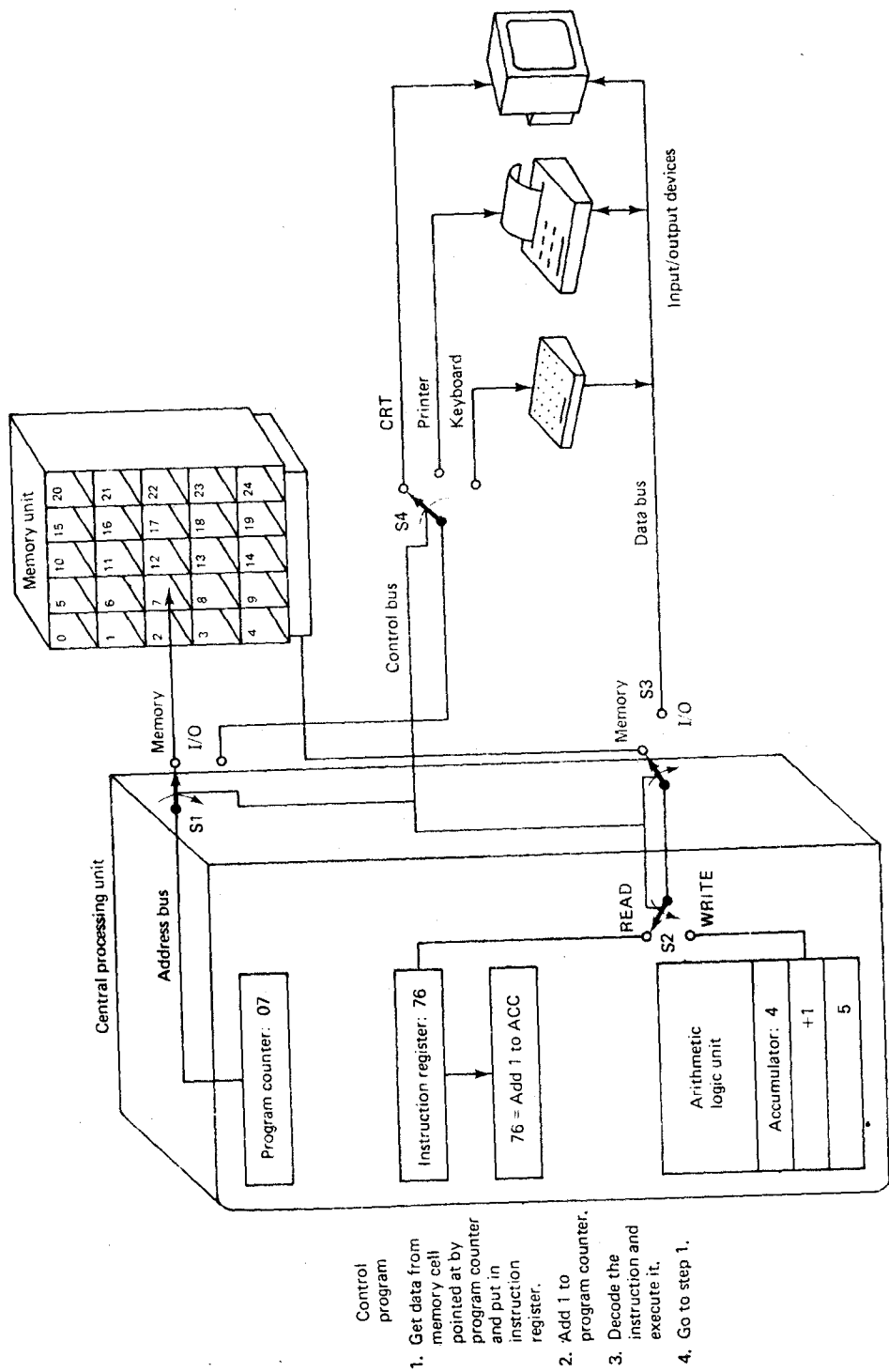


Figure 1.1 Block diagram of a digital computer. The three main blocks are the CPU (central processing unit), the memory unit, and the input/output (I/O) devices.

1. Fetch data from the memory cell whose address is currently in the program counter register. Put this data into the instruction register.
2. Add 1 to the address in the program counter.
3. Decode the command currently in the instruction register and do what it tells you.
4. Go to step 1.

These four steps constitute the principle of operation of all stored program digital computers. This includes the largest IBM mainframe to the tiniest microcomputer. The principle is called *fetch and execute* and is the key to understanding the activities of a microprocessor.

The Three-Bus Architecture. The CPU, memory unit, and I/O devices must be able to communicate with each other. For example, the CPU must be able to specify which memory cell is to be selected, and if the contents of that cell should be read or new data written into the cell.

This is the purpose of the address, data, and control buses shown in Fig. 1.1. When the CPU is required to read the contents of a particular memory cell, it first outputs the proper address on its *address bus*. This is actually the contents of the program counter. Next, the *control bus* causes switches S1 and S3 to switch to the MEMORY position and switch 2 to the READ position. The *data bus* now carries the contents of the selected memory cell back to the CPU and into the instruction register.

As a further example of the three-bus architecture, let's assume that the command in the instruction register requires the contents of the accumulator to be output to the printer. The execution of this command requires the CPU to output the address of the printer on its address bus. Switch S4 will examine this address, and seeing that it is for the printer, switch to the printer position. Next, the control bus will switch S1 and S3 to the I/O position and S2 to the WRITE position. The data to be output can now be placed on the data bus and routed to the printer.

In summary, the CPU begins each command cycle with an instruction fetch from the memory unit. The program counter is then incremented in preparation

TABLE 1.1. INSTRUCTION CYCLES OF A DIGITAL COMPUTER

Instruction type	Address bus	Control bus	Data bus
Memory read	Memory cell address	Select memory and read	Contents of selected memory cell
Memory write	Memory cell address	Select memory and write	Data to be written to memory
I/O read	I/O device address	Select I/O and read	Data from selected I/O device
I/O write	I/O device address	Select I/O and write	Data to be written to I/O device