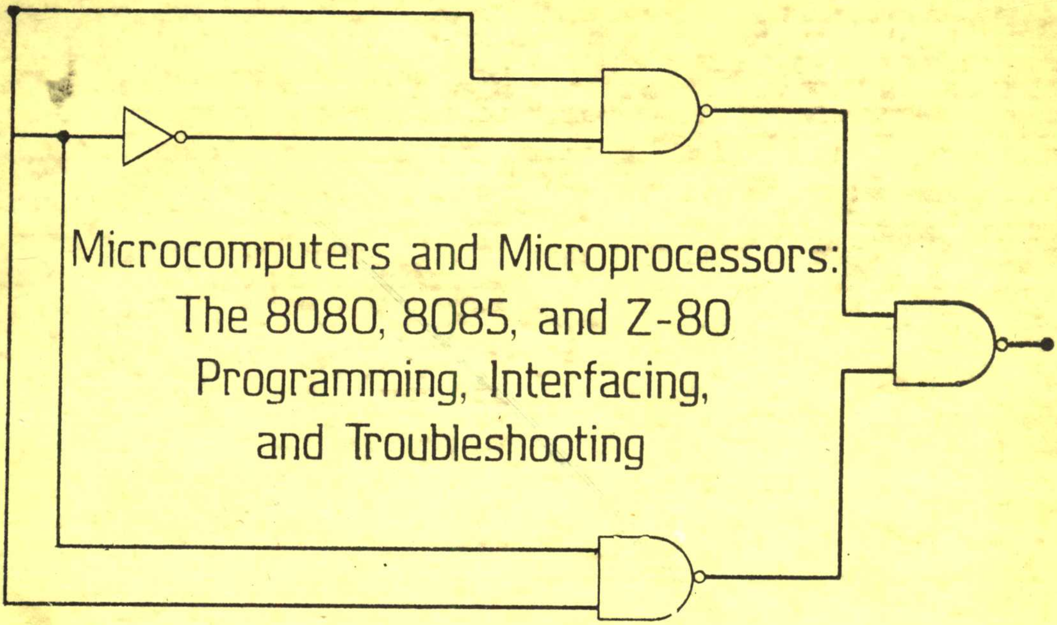
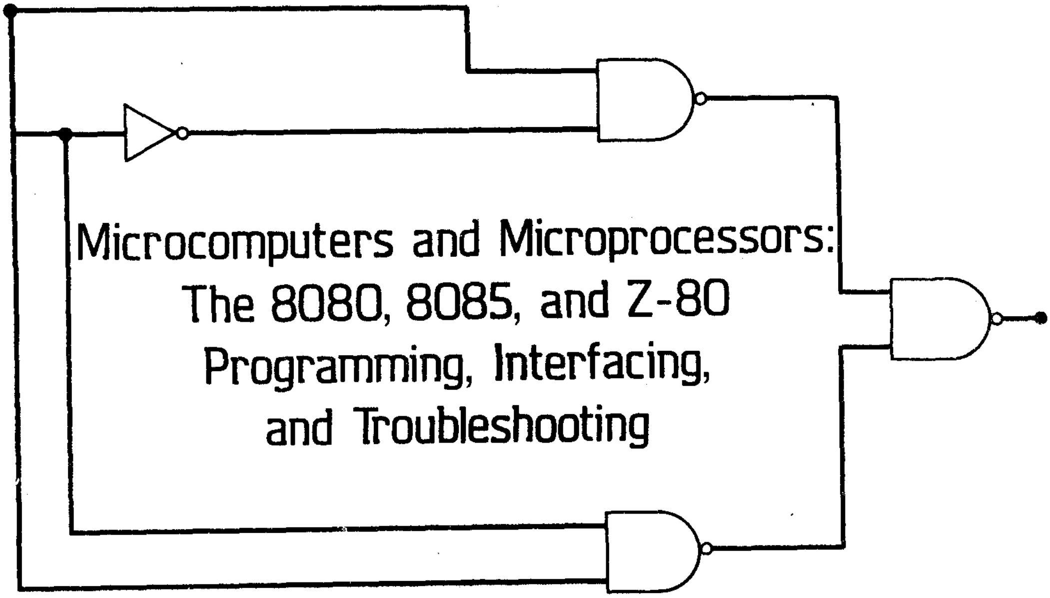


Microcomputers and Microprocessors: The 8080, 8085, and Z-80 Programming, Interfacing, and Troubleshooting

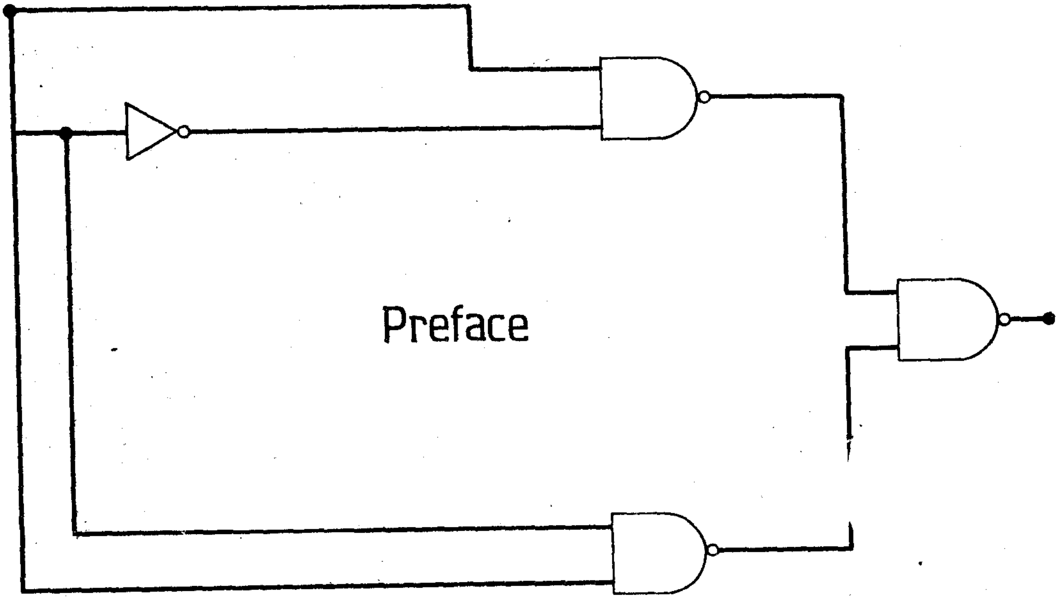
John Uffenbeck



Microcomputers and Microprocessors: The 8080, 8085, and Z-80 Programming, Interfacing, and Troubleshooting

John Uffenbeck

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



Preface

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.

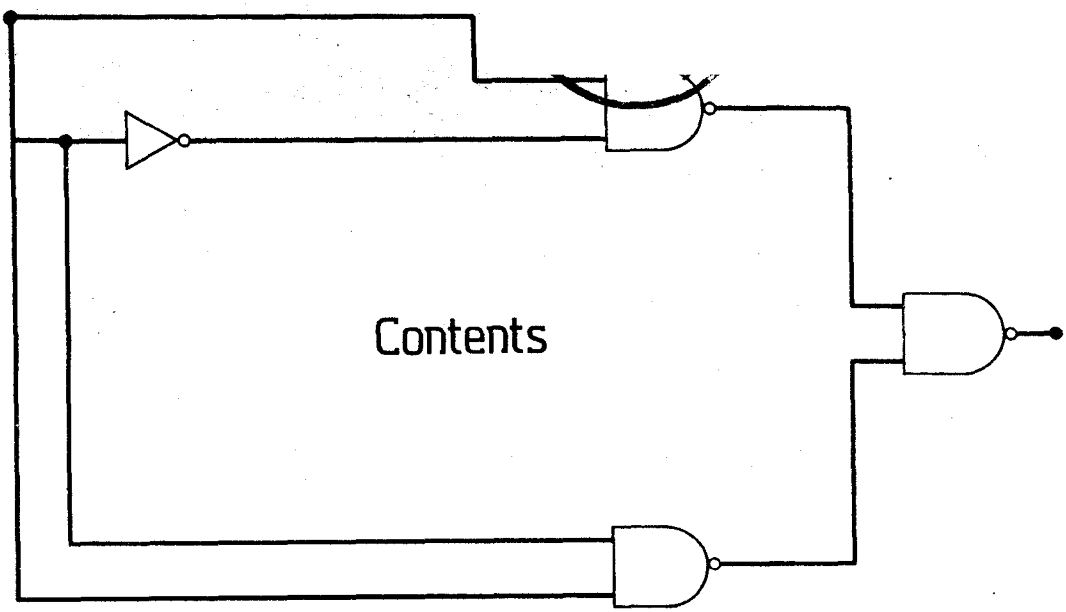
Recognizing that we must all learn to read the literature distributed by the semiconductor manufacturers, portions of over 40 data sheets are distributed throughout the text as appropriate. An appendix is also provided with handy references to hexadecimal instruction set summaries for the 8080, 8085,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

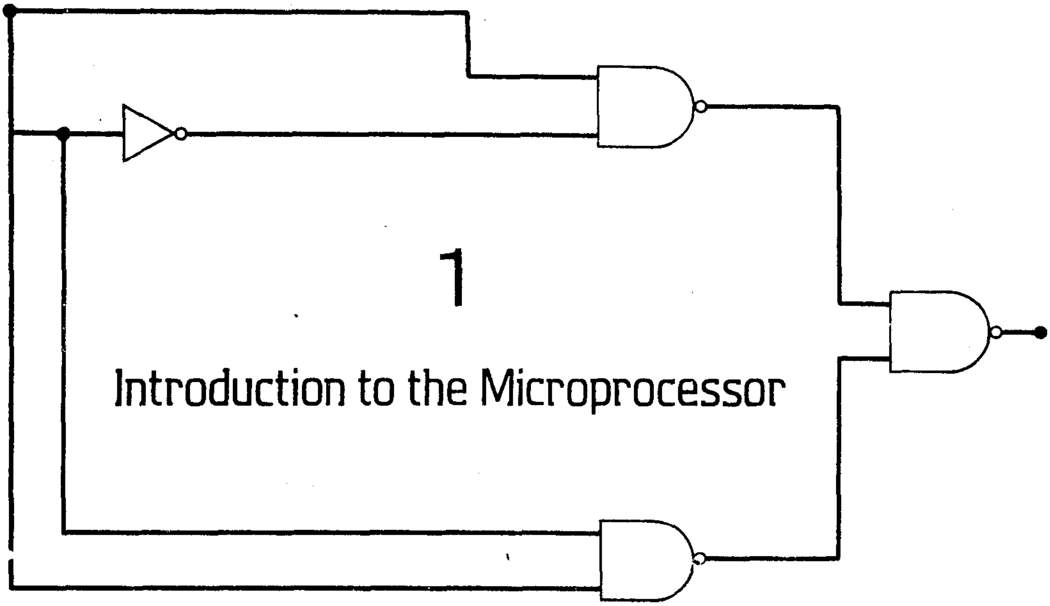
As a textbook, it can be used in a one- or two-semester course following the digital electronics sequence. Material that is unique to one of the three processors is usually highlighted so that readers interested in only one processor can skip inappropriate material. However, you may find yourself learning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



Contents



1

Introduction to the Microprocessor

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.

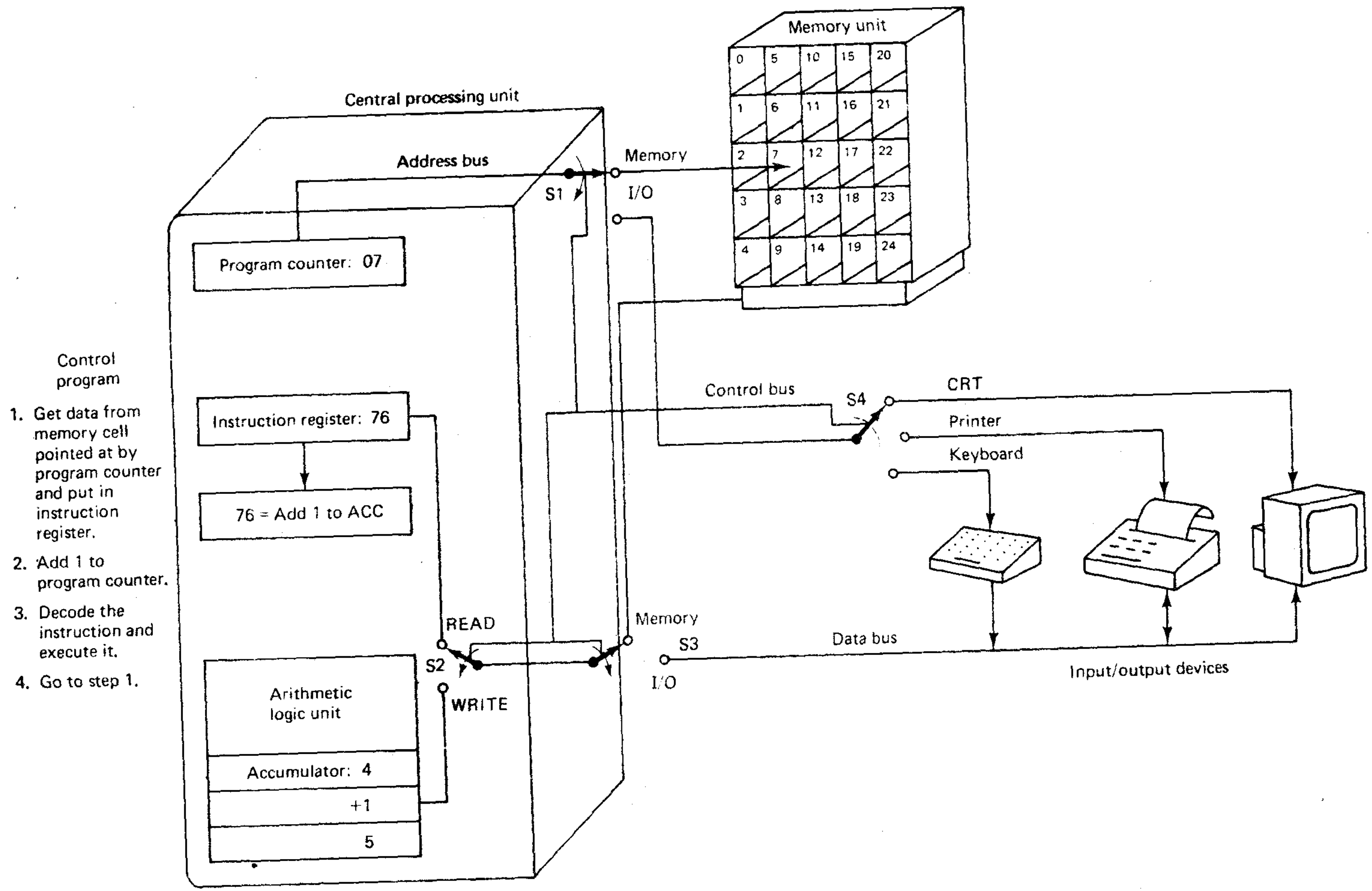


Figure 1.1 Block diagram of a digital computer. The three main blocks are the CPU (central processing unit), the memory unit, and the input/output (I/O) devices.

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