1

Microprocessors, Microcomputers, and Assembly Language

The microprocessor plays a significant role in the everyday functioning of industrialized societies. The microprocessor can be viewed as a programmable logic device that can be used to control processes or to turn on/off devices. On the other hand, the microprocessor can be viewed as a data processing unit or a computing unit of a computer. The **microprocessor** is a programmable integrated device that has computing and decision-making capability similar to that of the central processing unit (CPU) of a computer. Nowadays, the microprocessor is being used in a wide range of products called microprocessor-based products or systems. The microprocessor can be embedded in a larger system, can be a stand alone unit controlling processes, or it can function as the CPU of a computer called a **microcomputer.** This chapter introduces the basic structure of a microprocessor-based product and shows how the same structure is applicable to microcomputers and other large computers. The chapter concludes with an overview of microprocessor applications in the context of the entire spectrum of various computer applications and presents a block diagram of a temperature-control system as an application of the microprocessor-based system.

The microprocessor communicates and operates in the binary numbers 0 and 1, called **bits.** Each microprocessor has a fixed set of instructions in the form of binary patterns called a **machine language.** However, it is difficult for humans to communicate in the language of 0s and 1s. Therefore, the binary instructions are given abbreviated names, called **mnenomics,** which form the **assembly language** for a given microprocessor. This chapter explains both the machine language and the assembly language of the microprocessor, known as the 8085. The advantages of assembly language are compared with high-level languages (such as BASIC, C, C++, and Java).

OBJECTIVES

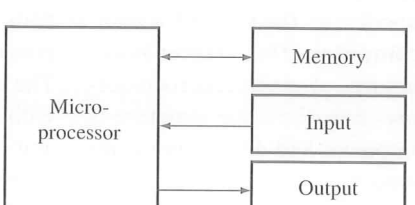
- ☐ Draw a block diagram of a microprocessor-based system and explain the functions of each component: microprocessor, memory, and I/O, and their lines of communication (the bus).
- ☐ Explain the terms *SSI, MSI,* and *LSI.*
- ☐ Define the terms *bit, byte, word, instruction, software,* and *hardware.*
- ☐ Explain the difference between the machine language and the assembly language of a computer.
- ☐ Explain the terms *low-level* and *high-level languages.*
- ☐ Explain the advantages of an assembly language over high-level languages.
- ☐ Define the term *ASCII* code and explain the relationship between the binary code and alphanumeric characters.
- ☐ Define the term *operating system.*
- ☐ List components and peripherals of a typical personal computer (PC).
- ☐ Draw a block diagram of a microprocessor-controlled temperature system (MCTS) and identify functions of each component.

1.1 MICROPROCESSORS

A microprocessor is a multipurpose, programmable, clock-driven, register-based electronic device that reads binary instructions from a storage device called *memory,* accepts binary data as input and processes data according to those instructions, and provides results as output. At a very elementary level, we can draw an analogy between microprocessor operations and the functions of a human brain that process information according to instructions (understanding) stored in its memory. The brain gets input from eyes and ears and sends processed information to output "devices" such as the face with its capacity to register expression, the hands or feet. However, there is no comparison between the complexity of a human brain and its memory and the relative simplicity of a microprocessor and its memory.

A typical programmable machine can be represented with four components: microprocessor, memory, input, and output, as shown in Figure 1.1. These four components work together or interact with each other to perform a given task; thus, they comprise a system. The physical components of this system are called **hardware.** A set of instructions written for the microprocessor to perform a task is called a **program,** and a group of programs is called **software.** The machine (system) represented in Figure 1.1 can be programmed to turn traffic lights on and off, compute mathematical functions, or keep track of a guidance system. This system may be simple or sophisticated, depending on its applications, and it is recognized by various names depending upon the purpose for which it is designed. The microprocessor applications are classified primarily in two categories: reprogrammable systems and embedded systems. In reprogrammable systems, such as microcomputers, the microprocessor is used for computing and data processing. These systems include

FIGURE 1.1
A Programmable Machine



general-purpose microprocessors capable of handling large data, mass storage devices (such as disks and CD-ROMs), and peripherals such as printers; a personal computer (PC) is a typical illustration. In embedded systems, the microprocessor is a part of a final product and is not available for reprogramming to the end user. A copying machine is a typical example of an embedded system. The microprocessors used in these systems are generally categorized as: (1) **microcontrollers** that include all the components shown in Figure 1.1 on one chip, and (2) general-purpose microprocessors with discrete components shown in Figure 1.1. Embedded systems can also be viewed as products that use microprocessors to perform their operations; they are known as microprocessor-based products. Examples include a wide range of products such as washing machines, dishwashers, automobile dashboard controls, traffic light controllers, and automatic testing instruments.

BINARY DIGITS

The microprocessor operates in binary digits, 0 and 1, also known as bits. **Bit** is an abbreviation for the term *binary digit.* These digits are represented in terms of electrical voltages in the machine: Generally, 0 represents one voltage level, and 1 represents another. The digits 0 and 1 are also synonymous with low and high, respectively.

Each microprocessor recognizes and processes a group of bits called the *word,* and microprocessors are classified according to their word length. For example, a processor with an 8-bit word is known as an 8-bit microprocessor, and a processor with a 32-bit word is known as a 32-bit microprocessor.

A MICROPROCESSOR AS A PROGRAMMABLE DEVICE

The fact that the microprocessor is programmable means it can be instructed to perform given tasks within its capability. A piano is a programmable machine; it is capable of generating various kinds of tones based on the number of keys it has. A musician selects keys depending upon the musical score printed on a sheet. Similarly, today's microprocessor is designed to understand and execute many binary instructions. It is a multipurpose machine: It can be used to perform various sophisticated computing functions, as well as simple tasks such as turning devices on or off. A programmer can select appropriate instructions and ask the microprocessor to perform various tasks on a given set of data.

The person who designs a piano determines the frequency (tone) for a given key and the scope of the piano music. Similarly, the engineers designing a microprocessor determine a set of tasks the microprocessor should perform and design the necessary logic circuits, and provide the user with a list of the instructions the processor will understand. For example, an instruction for adding two numbers may look like a group of eight binary digits, such as 1000 0000. These instructions are simply a pattern of 0s and 1s. The user (programmer) selects instructions from the list and determines the sequence of execution for a given task. These instructions are entered or stored in storage, called *memory,* which can be read by the microprocessor.

MEMORY

Memory is like the pages of a notebook with space for a fixed number of binary numbers on each line. However, these pages are generally made of semiconductor material. Typically, each line is an 8-bit register that can store eight binary bits, and several of these registers are

arranged in a sequence called memory. These registers are always grouped together in powers of two. For example, a group of $1024(2^{10})$ 8-bit registers on a semiconductor chip is known as 1K byte of memory; 1K is the closest approximation in thousands.* The user writes the necessary instructions and data in memory through an input device (described below), and asks the microprocessor to perform the given task and find an answer. The answer is generally displayed at an output device (described below) or stored in memory.

INPUT/OUTPUT

The user can enter instructions and data into memory through devices such as a keyboard or simple switches. These devices are called **input devices,** similar to eyes and ears in a human body. The microprocessor reads the instructions from the memory and processes the data according to those instructions. The result can be displayed by a device such as seven-segment LEDs (Light Emitting Diodes) or printed by a printer. These devices are called **output devices.**

MICROPROCESSOR AS A CPU (MPU)

We can also view the microprocessor as a primary component of a computer. Traditionally, the computer is represented in block diagram as shown in Figure 1.2(a). The block diagram shows that the computer has four components: memory, input, output, and the central processing unit (CPU), which consists of the arithmetic/logic unit (ALU) and the control unit. The CPU contains various registers to store data, the ALU to perform arithmetic and logical operations, instruction decoders, counters, and control lines. The CPU reads instructions from the memory and performs the tasks specified. It communicates with input/output devices either to accept or to send data. These devices are also known as peripherals. The CPU is the primary and central player in communicating with devices such as memory, input, and output. However, the timing of the communication process is controlled by the group of circuits called the **control unit.**

In the late 1960s, the CPU was designed with discrete components on various boards. With the advent of integrated circuit technology, it became possible to build the CPU on a single chip; this came to be known as a microprocessor, and the traditional block diagram shown in Figure 1.2(a) can be replaced by the block diagram shown in Figure 1.2(b). A computer with a microprocessor as its CPU is known as a microcomputer. The terms *microprocessor* and *microprocessor unit* (MPU) are often used synonymously. MPU implies a complete processing unit with the necessary control signals. Because of the limited number of available pins on a microprocessor package, some of the signals (such as control and multiplexed signals) need to be generated by using discrete devices to make the microprocessor a complete functional unit or MPU.

As semiconductor fabrication technology became more advanced, manufacturers were able to place not only MPU but also memory and I/O interfacing circuits on a single chip; this is known as a microcontroller or microcontroller unit (MCU). A microcontroller is essentially an entire computer on a single chip. Figure 1.2(c) shows that the microcontroller chip also includes additional devices such as an A/D converter, serial I/O, and timers (these devices are discussed in later chapters).

---

*In computer terminology, 1K is equal to 1024. In scientific terminology, 1k is equal to 1000.

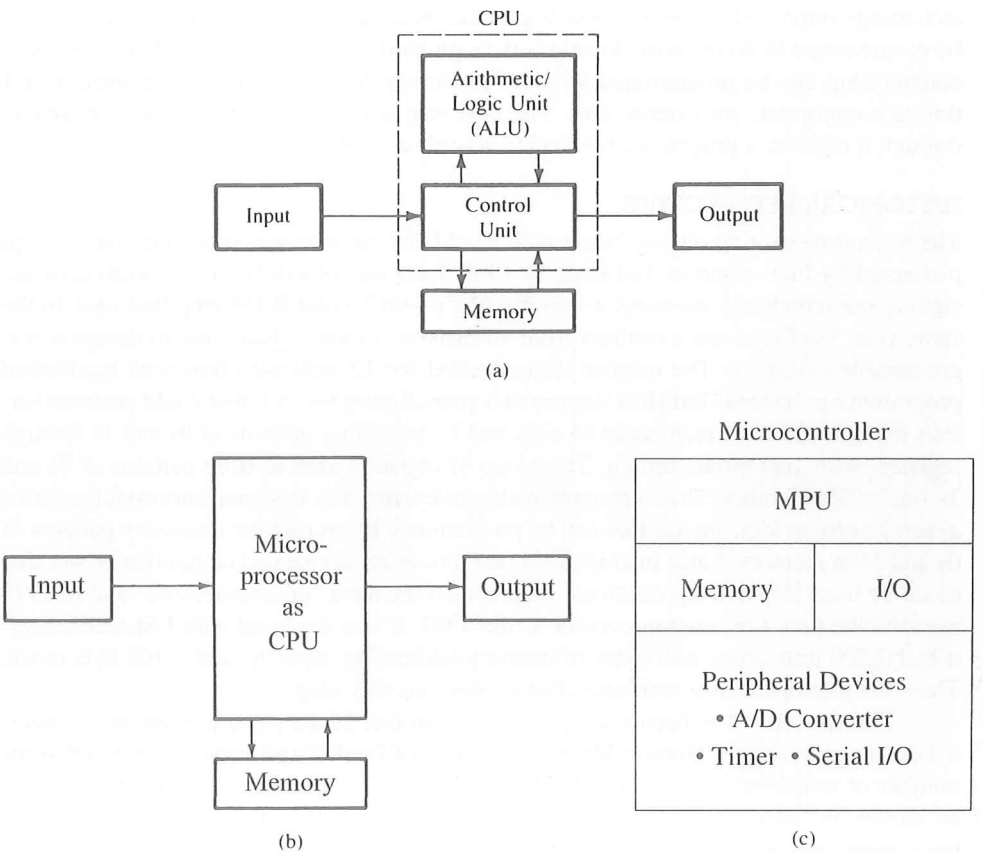


FIGURE 1.2
(a) Traditional Block Diagram of a Computer. (b) Block Diagram of a Computer with the Microprocessor as CPU; and (c) Block Diagram of a Microcontroller

1.1.1 Advances in Semiconductor Technology

Since 1950, semiconductor technology has undergone unprecedented changes. After the invention of the transistor, integrated circuits (ICs) appeared on the scene at the end of the 1950s; an entire circuit consisting of several transistors, diodes, and resistors could be designed on a single chip. In the early 1960s, logic gates known as the 7400 series were commonly available as ICs, and the technology of integrating the circuits of a logic gate on a single chip became known as small-scale integration (SSI). As semiconductor technology advanced, more than 100 gates were fabricated on one chip; this was called medium-scale integration (MSI). A typical example of MSI is a decade counter (7490). Within a few years, it was possible to fabricate more than 1000 gates on a single chip; this came to be known as large-scale integration (LSI). Now we are in the era of very-large-scale integration (VLSI) and super-large-scale integration (SLSI). The lines of demarcation between these different scales of integration are rather ill defined and arbitrary. As

technology improved, more and more logic circuits were built on one chip, and they could be programmed to do different functions through hard-wired connections. For example, a counter chip can be programmed to count in Hex or decimal by providing logic 0 or 1 through appropriate pin connections. The next step was the idea of providing 0s and 1s through a register, a programmable device described in the next section.

HISTORICAL PERSPECTIVE

The microprocessor revolution began with a bold and innovative approach in logic design pioneered by Intel engineer Ted Hoff. In 1969, Intel was primarily in the business of designing semiconductor memory; it introduced a 64-bit bipolar RAM chip that year. In the same year, Intel received a contract from a Japanese company, Busicom, to design a programmable calculator. The original design called for 12 different chips with hard-wired programming. Instead, Ted Hoff suggested a general-purpose chip that could perform various logic functions, which could be activated by providing patterns of 0s and 1s through registers with appropriate timing. The group of registers used to store patterns of 0s and 1s was called memory. Thus a programmable calculator was designed successfully with a general-purpose logic device that can be programmed by storing the necessary patterns of 0s and 1s in memory. Later Intel realized that this small device had computing power that could be used for many applications. Intel coined the term "microprocessor" and in 1971 released the first 4-bit microprocessor as the 4004. It was designed with LSI technology; it had 2,300 transistors, 640 bytes of memory-addressing capacity, and a 108 kHz clock. Thus, the microprocessor revolution began with this tiny chip.

This invention has been placed on a par with that of the printing press or the internal combustion engine. Gordon Moore, cofounder of Intel Corporation, predicted that the number of transistors per integrated circuit would double every 18 months; this came to be known as "Moore's Law." Just twenty-five years since the invention of the 4004, we have processors that are designed with 15 million transistors, that can address one terabyte (1×10^{12}) of memory, and that can operate at 400 MHz to 1.5-GHz frequency (see Table 1.1).

The Intel 4004 was quickly replaced by the 8-bit microprocessor (the Intel 8008), which was in turn superseded by the Intel 8080. In the mid-1970s, the Intel 8080 was widely used in control applications, and small computers also were designed using the 8080 as the CPU; these computers became known as microcomputers. Within a few years after the emergence of the 8080, the Motorola 6800, the Zilog Z80, and the Intel 8085 microprocessors were developed as improvements over the 8080. The 6800 was designed with a different architecture and the instruction set from the 8080. On the other hand, the 8085 and the Z80 were designed as **upward software compatible** with the 8080; that is, they included all the instructions of the 8080 plus additional instructions. As the microprocessors began to acquire more and more computing functions, they were viewed more as CPUs rather than as programmable logic devices. Most microcomputers are now built with 32- and 64-bit microprocessors. Each microprocessor has begun to carve a niche for its own applications. The 8-bit microprocessors are being used as programmable logic devices in control applications, and more powerful microprocessors are being used for mathematical computing (number crunching), data processing, and computer graphics applications.

TABLE 1.1
Intel Microprocessors: Historical Perspective

Processor	Year of Introduction	Number of Transistors	Initial Clock Speed	Address Bus	Data Bus	Addressable Memory
4004	1971	2,300	108 kHz	10-bit	4-bit	640 bytes
8008	1972	3,500	200 kHz	14-bit	8-bit	16 K
8080	1974	6,000	2 MHz	16-bit	8-bit	64 K
8085	1976	6,500	5 MHz	16-bit	8-bit	64 K
8086	1978	29,000	5 MHz	20-bit	16-bit	1 M
8088	1979	29,000	5 MHz	20-bit	8-bit*	1 M
80286	1982	134,000	8 MHz	24-bit	16-bit	16 M
80386	1985	275,000	16 MHz	32-bit	32-bit	4 G
80486	1989	1.2 M	25 MHz	32-bit	32-bit	4 G
Pentium	1993	3.1 M	60 MHz	32-bit	32/64-bit	4 G
Pentium Pro	1995	5.5 M	150 MHz	36-bit	32/64-bit	64 G
Pentium II	1997	8.8 M	233 MHz	36-bit	64-bit	64 G
Pentium III	1999	9.5 M	650 MHz	36-bit	64-bit	64 G
Pentium 4	2000	42 M	1.4 GHz	36-bit	64-bit	64 G

*External 8-bit and internal 16-bit data bus

Our focus here is on using 8-bit microprocessors as programmable devices in microprocessor-based systems. The overwhelming majority of microprocessor applications use 8-bit processors or microcontrollers. The range of applications is very wide and diversified—from the auto industry to home appliances. Recent statistics suggest that the worldwide sales of 8-bit chips range from 90 percent to 95 percent of the total number of microprocessor chips sold. However, the most compelling reason to use an 8-bit processor is educational. To understand the basic concepts underlying the microprocessor device, it is easier to learn from a simple 8-bit processor than from a 64-bit processor. And these fundamental concepts are easily transferable from 8-bit processors to larger processors.

1.1.2 Organization of a Microprocessor-Based System

Figure 1.3 shows a simplified but formal structure of a microprocessor-based system or a product. Since a microcomputer is one among many microprocessor-based systems, it will have the same structure as shown in Figure 1.3. It includes three components: *microprocessor, I/O (input/output),* and *memory* (read/write memory and read-only memory). These components are organized around a common communication path called a **bus.** The entire group of components is also referred to as a system or a microcomputer system, and the components themselves are referred to as sub-systems. At the outset, it is necessary to differentiate between the terms *microprocessor* and *microcomputer* because of the common misuse of these terms in popular literature. The microprocessor is one component of the microcomputer. On the other hand, the mi-

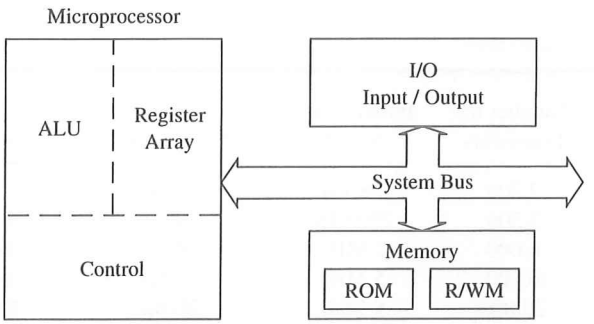


FIGURE 1.3
Microprocessor-Based System with Bus Architecture

crocomputer is a complete computer similar to any other computer, except that CPU functions of the microcomputer are performed by the microprocessor. Similarly, the term **peripheral** is used for input/output devices. The various components of a microprocessor-based product or a microcomputer are shown in Figure 1.3 and their functions are described in this section.

MICROPROCESSOR

The microprocessor is a clock-driven semiconductor device consisting of electronic logic circuits manufactured by using either a large-scale integration (LSI) or very-large-scale integration (VLSI) technique. The microprocessor is capable of performing various computing functions and making decisions to change the sequence of program execution. In large computers, a CPU implemented on one or more circuit boards performs these computing functions. The microprocessor is in many ways similar to the CPU, but includes all the logic circuitry, including the control unit, on one chip. The microprocessor can be divided into three segments for the sake of clarity, as shown in Figure 1.3: arithmetic/logic unit (ALU), register array, and control unit.

Arithmetic/Logic Unit This is the area of the microprocessor where various computing functions are performed on data. The ALU unit performs such arithmetic operations as addition and subtraction, and such logic operations as AND, OR, and exclusive OR.

Register Array This area of the microprocessor consists of various registers identified by letters such as B, C, D, E, H, and L. These registers are primarily used to store data temporarily during the execution of a program and are accessible to the user through instructions.

Control Unit The control unit provides the necessary timing and control signals to all the operations in the microcomputer. It controls the flow of data between the microprocessor and memory and peripherals.

Now the question is: What is the relationship among the programmer's instruction (binary pattern of 0s and 1s), the ALU, and the control unit? This can be explained with the example of a full adder circuit. A full adder circuit can be designed with registers, logic gates, and a clock. The clock initiates the adding operation. Similarly, the bit pattern of an instruction initiates a sequence of clock signals, activates the appropriate logic circuits in the ALU, and performs the task. This is called microprogramming, which is done in the design stage of the microprocessor. In many ways, this is similar to the process of how our brain operates. In early childhood, we learn a word, "sit," and physical motions needed for the action are embedded in our brain. When we hear the word "sit," our brain activates a series of actions for our muscles and bones and we sit down. In this analogy, the word "sit" is like an instruction in a microprocessor, and actions initiated by our brain are like microprograms.

The bit patterns required to initiate these microprogram operations are given to the programmer in the form of the instruction set of the microprocessor. The programmer selects appropriate bit patterns from the set for a given task and enters them sequentially in memory through an input device. When the CPU reads these bit patterns one at a time, it initiates appropriate microprograms through the control unit, and performs the task specified in the instructions.

At present, various microprocessors are available from different manufacturers. Examples of widely used 8-bit microprocessors include the Intel 8085, Zilog Z80, and Motorola 68008. Earlier microcomputers were designed around the 8-bit microprocessors; now these processors are generally used in embedded systems. The recent versions of IBM personal computers are designed around the Intel 32- or 64-bit microprocessors. Single-board microcomputers such as the SDK-85 (Intel), The Primer (EMAC Inc.), and the Micro-Professor (Multitech) are commonly used in college laboratories; the SDK-85 and The Primer (described in Appendix B) are based on the 8085 microprocessor, and the Micro-Professor is based on the Z80 microprocessor.

MEMORY

Memory stores such binary information as instructions and data, and provides that information to the microprocessor whenever necessary. To execute programs, the microprocessor reads instructions and data from memory and performs the computing operations in its ALU section. Results are either transferred to the output section for display or stored in memory for later use. The memory block shown in Figure 1.3 has two sections: **Read-Only memory** (ROM) and **Read/Write memory** (R/WM), popularly known as **Random-Access memory** (RAM).

The ROM is used to store programs that do not need alterations. The monitor program of a single-board microcomputer is generally stored in the ROM. This program interprets the information entered through a keyboard and provides equivalent binary digits to the microprocessor. Programs stored in the ROM can only be read; they cannot be altered.

The Read/Write memory (R/WM) is also known as *user memory.* It is used to store user programs and data. In single-board microcomputers, the monitor program monitors the Hex keys and stores those instructions and data in the R/W memory. The information stored in this memory can be easily read and altered.

I/O (INPUT/OUTPUT)

The third component of a microprocessor-based system is I/O (input/output); it communicates with the outside world. I/O includes two types of devices: input and output; these I/O devices are also known as *peripherals.*

The input devices such as a keyboard, switches, and an analog-to-digital (A/D) converter transfer binary information (data and instructions) from the outside world to the microprocessor. Typically, a microcomputer used in college laboratories includes either a hexadecimal keyboard or an ASCII keyboard as an input device. The hexadecimal (Hex) keyboard has 16 data keys (0 to 9 and A to F) and some additional function keys to perform such operations as storing data and executing programs. The ASCII (the term is explained in Section 1.2) keyboard is similar to a typewriter keyboard, and it is used to enter programs in an English-like language. Although the ASCII keyboard is found in most microcomputers (PCs), single-board microcomputers generally have Hex keyboards, and microprocessor-based products such as a microwave oven have decimal keyboards.

The output devices transfer data from the microprocessor to the outside world. They include devices such as light emitting diodes (LEDs), a cathode-ray tube (CRT) or video screen, a printer, X-Y plotter, a magnetic tape, and digital-to-analog (D/A) converter. Typically, single-board microcomputers and microprocessor-based products (such as a dishwasher or a microwave oven) include LEDs, seven-segment LEDs, and alphanumeric LED displays as output devices. Microcomputers (PCs) are generally equipped with output devices such as a video screen (also called a monitor) and a printer.

SYSTEM BUS

The system bus is a communication path between the microprocessor and peripherals; it is nothing but a group of wires to carry bits. In fact, there are several buses in the system that will be discussed in the next chapter. All peripherals (and memory) share the same bus; however, the microprocessor communicates with only one peripheral at a time. The timing is provided by the control unit of the microprocessor.

1.1.3 How Does the Microprocessor Work?

Assume that a program and data are already entered in the R/W memory. (How to write and execute a program will be explained later.) The program includes binary instructions to add given data and to display the answer at the seven-segment LEDs. When the microprocessor is given a command to execute the program, it reads and executes one instruction at a time and finally sends the result to the seven-segment LEDs for display.

This process of program execution can best be described by comparing it to the process of assembling a radio kit. The instructions for assembling the radio are printed in a sequence on a sheet of paper. One reads the first instruction, then picks up the necessary components of the radio and performs the task. The sequence of the process is *read, interpret,* and *perform.* The microprocessor works the same way. The instructions are stored sequentially in the memory. The microprocessor fetches the first instruction

from its memory sheet, decodes it, and executes that instruction. The sequence of *fetch, decode,* and *execute* is continued until the microprocessor comes across an instruction to *stop.* During the entire process, the microprocessor uses the system bus to fetch the binary instructions and data from the memory. It uses registers from the register section to store data temporarily, and it performs the computing function in the ALU section. Finally, it sends out the result in binary, using the same bus lines, to the seven-segment LEDs.

1.1.4 Summary of Important Concepts

The functions of various components of a microprocessor-based system can be summarized as follows:

1. The microprocessor
   - ☐ reads instructions from memory.
   - ☐ communicates with all peripherals (memory and I/Os) using the system bus.
   - ☐ controls the timing of information flow.
   - ☐ performs the computing tasks specified in a program.
2. The memory
   - ☐ stores binary information, called instructions and data.
   - ☐ provides the instructions and data to the microprocessor on request.
   - ☐ stores results and data for the microprocessor.
3. The input device
   - ☐ enters data and instructions under the control of a program such as a monitor program.
4. The output device
   - ☐ accepts data from the microprocessor as specified in a program.
5. The bus
   - ☐ carries bits between the microprocessor and memory and I/Os.

1.2 MICROPROCESSOR INSTRUCTION SET AND COMPUTER LANGUAGES

Microprocessors recognize and operate in binary numbers. However, each microprocessor has its own binary words, meanings, and language. The words are formed by combining a number of bits for a given machine. The **word** (or word length) is defined as the number of bits the microprocessor recognizes and processes at a time. The word length ranges from four bits for small, microprocessor-based systems to 64 bits for high-speed large computers. Another term commonly used to express word length is byte. A **byte** is defined as a group of eight bits. For example, a 16-bit microprocessor has a word length equal to two bytes. The term **nibble,** which stands for a group of four bits, is found also in popular computer magazines and books. A byte has two nibbles.

Each machine has its own set of instructions based on the design of its CPU or of its microprocessor. To communicate with the computer, one must give instructions in binary language **(machine language).** Because it is difficult for most people to write programs in sets of 0s and 1s, computer manufacturers have devised English-like words to represent the binary instructions of a machine. Programmers can write programs, called **assembly language** programs, using these words. Because an assembly language is specific to a given machine, programs written in assembly language are not transferable from one machine to another. To circumvent this limitation, such general-purpose languages as BASIC and FORTRAN have been devised; a program written in these languages can be machine-independent. These languages are called **high-level languages.** This section deals with various aspects of these three types of languages: machine, assembly, and high-level. The machine and assembly languages are discussed in the context of the 8085 microprocessor.

1.2.1 Machine Language

The number of bits in a word for a given machine is fixed, and words are formed through various combinations of these bits. For example, a machine with a word length of eight bits can have 256 (2^8) combinations of eight bits—thus a language of 256 words. However, not all of these words need to be used in the machine. The microprocessor design engineer selects combinations of bit patterns and gives a specific meaning to each combination by using electronic logic circuits; this is called an **instruction.** Instructions are made up of one word or several words. The set of instructions designed into the machine makes up its machine language—a binary language, composed of 0s and 1s—that is specific to each computer. In this book, we are concerned with the language of a widely used 8-bit microprocessor, the 8085, manufactured by Intel Corporation. The primary focus here is on the microprocessor because the microprocessor determines the machine language and the operations of a microprocessor-based system.

1.2.2 8085 Machine Language

The 8085 is a microprocessor with 8-bit word length: its **instruction set** (or language) is designed by using various combinations of these eight bits. The 8085 is an improved version of the earlier processor 8080A.

An *instruction* is a binary pattern entered through an input device in memory to command the microprocessor to perform that specific function.

For example:

0011 1100	is an instruction that increments the number in the register called the **accumulator** by one.
1000 0000	is an instruction that adds the number in the register called B to the number in the accumulator, and keeps the sum in the accumulator.

The 8085 microprocessor has 246 such bit patterns, amounting to 74 different instructions for performing various operations. These 74 different instructions are called its instruction set. This binary language with a predetermined instruction set is called the 8085 machine language.

Because it is tedious and error-inducive for people to recognize and write instructions in binary language, these instructions are, for convenience, written in hexadecimal code and entered in a single-board microcomputer by using Hex keys. For example, the binary instruction 0011 1100 (mentioned previously) is equivalent to 3C in hexadecimal. This instruction can be entered in a single-board microcomputer system with a Hex keyboard by pressing two keys: 3 and C. The monitor program of the system translates these keys into their equivalent binary pattern.

1.2.3 8085 Assembly Language

Even though the instructions can be written in hexadecimal code, it is still difficult to understand a program written in hexadecimal numbers. Therefore, each manufacturer of a microprocessor has devised a symbolic code for each instruction, called a **mnemonic.** (The word *mnemonic* is based on the Greek word meaning *mindful;* that is, a memory aid.) The mnemonic for a particular instruction consists of letters that suggest the operation to be performed by that instruction.

For example, the binary code 0011 1100 ($3C_{16}$ or 3CH* in hexadecimal) of the 8085 microprocessor is represented by the mnemonic INR A:

INR A — INR stands for increment, and A represents the accumulator. This symbol suggests the operation of incrementing the accumulator contents by one.

Similarly, the binary code 1000 0000 (80_{16} or 80H) is represented as

ADD B — ADD stands for addition, and B represents the contents in register B. This symbol suggests the addition of the contents in register B and the contents in the accumulator.

Although these symbols do not specify the complete operations, they suggest its significant part. The complete description of each instruction must be supplied by the manufacturer. The complete set of 8085 mnemonics is called the 8085 assembly language, and a program written in these mnemonics is called an assembly language program. (Again, the assembly language, or mnemonics, is specific to each microprocessor. For example, the Motorola 6800 microprocessor has an entirely different set of binary codes and mnemonics than the 8085. Therefore, the assembly language of the 6800 is far different from that of the 8085.) An assembly language program written for one microprocessor is not transferable to a computer with another microprocessor unless the two microprocessors are compatible in their machine codes.

Machine language and assembly language are microprocessor-specific and are both considered **low-level languages.** The machine language is in binary, and the assembly language is in English-like words; however, the microprocessor understands only the binary. How, then, are the assembly language mnemonics written and translated into machine language or binary code? The mnemonics can be written by hand on paper (or in a

*Hexadecimal numbers are shown either with the subscript 16, or as a number followed by the letter H.

notebook) and translated manually in hexadecimal code, called **hand assembly,** as explained in Section 1.25. Similarly, the mnemonics can be written electronically on a computer using a program called an Editor in the ASCII code (explained in the next section) and translated into binary code by using the program called an **assembler.**

1.2.4 ASCII Code

A computer is a binary machine; to communicate with the computer in alphabetic letters and decimal numbers, translation codes are necessary. The commonly used code is known as **ASCII**—American Standard Code for Information Interchange. It is a 7-bit code with 128 (2^7) combinations, and each combination from 00H to 7FH is assigned to either a letter, a decimal number, a symbol, or a machine command (see Appendix E). For example, hexadecimal 30H to 39H represent 0 to 9 decimal digits, 41H to 5AH represent capital letters A through Z, 20H to 2FH represent various symbols, and initial codes 00H to 1FH represent such machine commands as carriage return and line feed. In microcomputer systems, keyboards (called ASCII keyboards), video screens, and printers are typical examples of devices that use ASCII codes. When the key "9" is pressed on an ASCII keyboard, the computer receives 39H in binary, called an ASCII character, and the system program translates ASCII characters into appropriate binary numbers.

However, recent computers use many more characters than the original 128 combinations; this is called Extended ASCII. It is an 8-bit code that provides 256 (2^8) combinations; the additional 128 combinations are assigned to various graphics characters.

1.2.5 Writing and Executing an Assembly Language Program

As we explained earlier, a program is a set of logically related instructions written in a specific sequence to accomplish a task. To manually write and execute an assembly language program on a single-board computer, with a Hex keyboard for input and LEDs for output, the following steps are necessary:

1. Write the instructions in mnemonics obtained from the instruction set supplied by the manufacturer.
2. Find the hexadecimal machine code for each instruction by searching through the set of instructions.
3. Enter (load) the program in the user memory in a sequential order by using the Hex keyboard as the input device.
4. Execute the program by pressing the Execute key. The answer will be displayed by the LEDs.

This procedure is called either **manual** or **hand assembly.**

When the user program is entered by the keys, each entry is interpreted and converted into its binary equivalent by the monitor program, and the machine code is stored as eight bits in each memory location in a sequence. When the Execute command is given, the microprocessor fetches each instruction, decodes it, and executes it in a sequence until the end of the program.

The manual assembly procedure is commonly used in single-board microcomputers and is suited for small programs; however, looking up the machine codes and entering the program is tedious and subject to errors. The other process involves the use of a computer with an ASCII keyboard and an assembler.

The **assembler** is a program that translates the mnemonics entered by the ASCII keyboard into the corresponding binary machine codes of the microprocessor. Each microprocessor has its own assembler because the mnemonics and machine codes are specific to the microprocessor being used, and each assembler has rules that must be followed by the programmer. Personal Computers (PCs—see Section 1.3) are commonly available on college campuses. These computers are based on 16- or 32-bit microprocessors with different mnemonics than the 8085 microprocessor. However, the programs known as **cross-assemblers** can be used to translate the 8085 mnemonics into appropriate machine codes. (Assemblers and cross-assemblers are discussed in Chapter 11.)

1.2.6 High-Level Languages

Programming languages that are intended to be machine-independent are called **high-level languages.** These include such languages as BASIC, PASCAL, C, C++, and Java, all of which have certain sets of rules and draw on symbols and conventions from English. Instructions written in these languages are known as **statements** rather than mnemonics. A program written in BASIC for a microcomputer with the 8085 microprocessor can generally be run on another microcomputer with a different microprocessor.

Now the question is: How are words in English converted into the binary languages of different microprocessors? The answer is: Through another program called either a **compiler** or an **interpreter.** These programs accept English-like statements as their input, called the *source code.* The compiler or interpreter then translates the source code into the machine language compatible with the microprocessor being used in the system. This translation in the machine language is called the *object code* (Figure 1.4). Each microprocessor needs its own compiler or an interpreter for each high-level language. The primary difference between a compiler and an interpreter lies in the process of generating machine code. The compiler reads the entire program first and translates it into the object code that is executed by the microprocessor. On the other hand, the interpreter reads one instruction at a time, produces its object code (a sequence of machine actions), and executes the instruction before reading the next instruction. M-Basic is a common example of an interpreter for BASIC language. Compilers are generally used in such languages as FORTRAN, PASCAL, C, and C++.

Compilers and interpreters require large memory space because an instruction in English requires several machine codes to translate it into binary. On the other hand, there is one-to-one correspondence between the assembly language mnemonics and the ma-

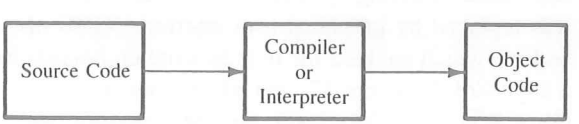


FIGURE 1.4
Block Diagram: Translation of High-Level Language Program into Machine Code

chine code. Thus, assembly language programs are compact and require less memory space. They are more efficient than the high-level language programs. The primary advantage of high-level languages is in troubleshooting **(debugging)** programs. It is much easier to find errors in a program written in a high-level language than to find them in a program written in an assembly language.

In certain applications such as traffic control and appliance control, where programs are small and compact, assembly language is suitable. Similarly, in such real-time applications as converting a high-frequency waveform into digital data, program efficiency is critical. In real-time applications, events and time should closely match each other without significant delay; therefore, assembly language is highly desirable in these applications. On the other hand, for applications in which programs are large and memory is not a limitation, high-level languages may be desirable. Typical examples of applications programs are word processors, video games, tax-return preparation, billing, accounting, and money management. These programs are generally written by professionals such as programmers in high-level languages. The advantage of time saved in debugging a large program may outweigh the disadvantages of memory requirements and inefficiency. Now we need to examine the relationship and the interaction between the hardware (microprocessor, memory, and I/O) and software (languages and application programs).

1.2.7 Operating Systems

The interaction between the hardware and the software is managed by a set of programs called an **operating system** of a computer; it oversees all the operations of the computer. The computer transfers information constantly between memory and various peripherals such as printer, keyboard, and video monitor. It also stores programs on disk. The operating system is responsible primarily for storing information on the disk and for the communication between the computer and its peripherals. The functional relationship between the operating system and the hardware of the computer is shown in Figure 1.5(a).

Figure 1.5(b) shows the relationship and the hierarchy among the hardware, the operating system, high-level languages, and application programs. The operating system is closest to the hardware and application programs are farthest from the hardware. When the computer is turned on, the operating system is in charge of the system and stays in the background and provides channels of communications to application programs. Each computer has its own operating system. In the 1970s, CP/M (Control Monitor Program) was a widely used operating system; it was designed for 8-bit processors such as the Z80 and 8085/8080A. In the 1980s, when 8-bit processors were replaced by 16-bit processors in personal computers (PCs), MS-DOS (Microsoft Disk Operating System) replaced CP/M. MS-DOS (also known as DOS or PC-DOS) was designed to handle 16-bit processor systems, and it became almost an industry standard for the personal computer. MS-DOS is a text-based operating system; the commands are written using a keyboard. In the 1990s, it was replaced by graphical user interface (GUI) operating systems such as Windows 3.1 and 95, which enabled the user to write commands by clicking on icons rather than using a keyboard. In recent 32- and 64-bit computers, operating systems such as UNIX, Linux, OS/2, Windows 95/98/2000, ME (Millennium), and NT are commonly used.