3.1 THE DIGITAL COMPUTER

A digital computer is a collection of logic elements that can execute arbitrary algorithms to perform data calculation and manipulation functions. A computer is composed of a microprocessor, memory, and some input/output (I/O) elements as shown in Fig. 3.1. The microprocessor, often called a microprocessor unit (MPU) or central processing unit (CPU), contains logic to step through an algorithm, called a *program*, that has been stored in the computer's program memory. The data used and manipulated by that program is held in the computer's data memory. Memory is a repository for data that is usually organized as a linear array of individually accessible locations. The microprocessor can access a particular location in memory by presenting a memory address (the index of the desired location) to the memory element. I/O elements enable the microprocessor to communicate with the outside world to acquire new data and present the results of its programmed computations. Such elements can include a keyboard or display controller.

Programs are composed of many very simple individual operations, called *instructions,* that specify in exact detail how the microprocessor should carry out an algorithm. A simple program may have dozens of instructions, whereas a complex program can have tens of millions of instructions. Collectively, the programs that run on microprocessors are called *software*, in contrast to the *hardware* on which they run. Each type of microprocessor has its own *instruction set* that defines the full set of unique, discrete operations that it is capable of executing. These instructions perform very narrow tasks that, on their own, may seem insignificant. However, when thousands or millions of these tiny instructions are strung together, they may create a video game or a word processor.

A microprocessor possesses no inherent intelligence or capability to spontaneously begin performing useful work. Each microprocessor is constructed with an instruction set that can be invoked in arbitrary sequences. Therefore, a microprocessor has the potential to perform useful work but will do nothing of the sort on its own. To make the microprocessor perform useful work, it requires explicit guidance in the form of software programming. A task of even moderate complexity must be broken down into many tiny steps to be implemented on a microprocessor. These steps include basic arithmetic, Boolean operations, loading data from memory or an input element such as a keyboard, and storing data back to memory or an output element such as a printer.

Memory structure is one of a computer's key characteristics, because the microprocessor is almost constantly accessing it to retrieve a new instruction, load new data to operate on, or store a calculated result. While program and data memory are logically distinct classifications, they may share the same physical memory resource. *Random access memory* (RAM) is the term used to describe a generic memory resource whose locations can be accessed, or *addressed*, in an arbitrary order and either read or written. A *read* is the process of retrieving data from a memory address and loading it into the microprocessor. A *write* is the process of storing data to a memory address from the microprocessor. Both programs and data can occupy RAM. Consider your desktop computer. When you

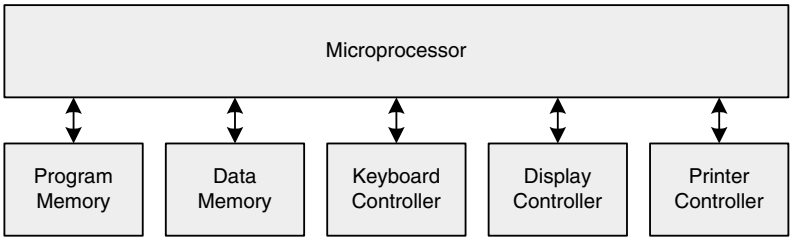

FIGURE 3.1 Generic computer block diagram.

execute a program that is located on the disk drive, that program is first loaded into the computer's RAM and then executed from a region set aside for program memory. As on a desktop computer, RAM is most often *volatile*—meaning that it loses its contents when the power is turned off.

Some software cannot be stored in volatile memory, because basic initialization instructions, or *boot code*, must be present when the computer is turned on. Remember that a microprocessor can do nothing useful without software being readily available. When power is first applied to a computer, the microprocessor must be able to quickly locate boot code so that it can get itself ready to accept input from a user or load a program from an input device. This startup sequence is called *booting*, hence the term *boot code*. When you turn your computer on, the first messages that it displays on the monitor are a product of its boot code. Eventually, the computer is able to access its disk drive and begins loading software into RAM as part of its normal operation. To ensure that boot code is ready at power-up, *nonvolatile* memory called *read only memory* (ROM) exists. ROM can be used to store both programs as well as any data that must be present at power-up and immediately accessible. Software contained in ROM is also known as *firmware*. As its name implies, ROM can only be read but not written. More complex computers contain a relatively small quantity of ROM to hold basic boot code that then loads main operating software from another device into RAM. Small computers may contain all of their software in ROM. Figure 3.2 shows how ROM and RAM complement each other in a typical computer architecture.

A microprocessor connects to devices such as memory and I/O via *data* and *address buses*. Collectively, these two buses can be referred to as the *microprocessor bus*. A bus is a collection of wires that serve a common purpose. The data bus is a bit array of sufficient size to communicate one complete data unit at a time. Most often, the data bus is one or more bytes in width. An eight-bit microprocessor, operating on one byte at time, almost always has an eight-bit data bus. A 32-bit microprocessor, capable of operating on up to 4 bytes at a time, can have a data bus that is 32, 16, or 8 bits wide. The exact data bus width is implementation specific and varies according to the intended application of the microprocessor. A narrower bus width means that it will take more time to communicate a quantity of data as compared to a wider bus. Common notation for a data bus is *D[7:0]* for an 8-bit bus and *D[31:0]* for a 32-bit bus, where 0 is the least-significant bit.

The address bus is a bit array of sufficient size to fully express the microprocessor's *address space*. Address space refers to the maximum amount of memory and I/O that a microprocessor can directly address. If a microprocessor has a 16-bit address bus, it can address up to 2^{16} = 65,536 bytes. Therefore, it has a 64 kB address space. The entire address space does not have to be used; it simply establishes a maximum limit on memory size. Common notation for a 16-bit address bus is *A[15:0]*, where 0 is the least-significant bit. Figure 3.3 shows a typical microprocessor bus configuration in a computer. Note that the address bus is unidirectional (the microprocessor asserts requested addresses to the various devices), and the data bus is bidirectional (the microprocessor asserts data on a write and the devices assert data on reads).

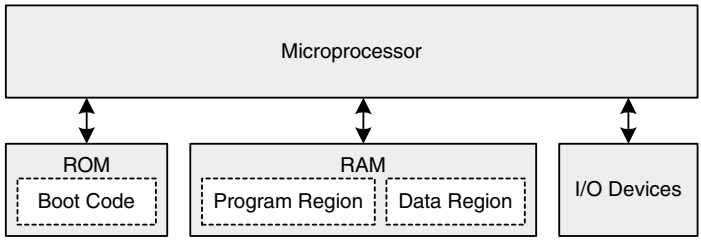

FIGURE 3.2 Basic ROM/RAM memory complement.