to be processed or executed within the machine (see Figure 10.2). Sequences of individual computer instructions make up a computer program, which is the set of commands defining what a computer does at any given time.

In this chapter, we will examine computer architecture—that is, we will learn how computer hardware is organized. We will also see how computers do many fascinating tasks simply by manipulating ones and zeros.

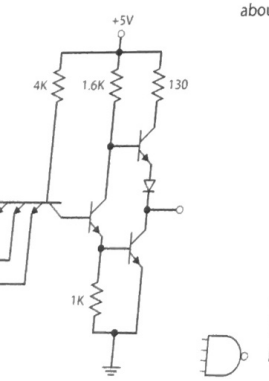
SECTION 1: Digital Logic

KEY IDEAS >

- Basic digital logic functions are the building blocks of computer hardware design.
- Digital system design is based on the binary number system.
- Computer hardware is composed of key architectural elements including registers, counters, and ALUs.

Digital logic devices are used to create digital systems, including computers, MP3 players, high-definition TVs, and many other products. Digital logic devices are specialized electrical circuits built from transistors, diodes, and resistors. By arranging these electrical components in various ways, logic functions are created. Logic functions determine how digital signals are processed. When logic functions are combined, they produce digital systems. Basic digital logic functions are the building blocks of computer hardware design.

The computer is a common digital system and is the primary subject of this chapter. In order to understand how computers are designed, we need to learn about various common logic devices and about the binary number system.



Digital devices, such as this 4-input NAND gate, are constructed from electronic components.

Logic Families

Digital logic devices are useful in digital design because they exist in logic families. A logic family is a group of digital circuits with similar electrical properties. Logic families may contain thousands of specialized logic devices. Digital system designers like working with logic families since they can concentrate their design efforts on the logical design of a system rather than on the electrical design. By eliminating the need to consider the voltages and currents in a circuit, for example, the designer is free to interconnect devices at the logic level to build the digital system desired. Manufacturers consider electrical requirements when they create logic families, giving the digital designer the freedom to treat digital devices as interconnecting building blocks. Figure 10.3 shows how various electronic parts are connected to create a logic device.

CMOS (complementary metal oxide semiconductor) is one circuit configuration used in constructing logic devices. Another widely used circuit form is TTL (transistor-transistor logic). CMOS and TTL logic families have been available for many years. Each has unique electrical characteristics, and both create similar logic devices or functions. When designers interconnect CMOS devices, for example, each device understands the electrical signals sent to it from other CMOS devices. Each device understands the signals from others in an electrical sense; thus, logic design becomes relatively easy.

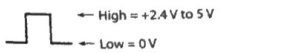


Figure 10.4 | Logic levels in the TTL system consist of two distinct ranges of voltages.

Voltage signals in digital systems are far different from voltage signals in a radio or a table lamp. Many analog systems use a range of voltages to function, but digital systems use only two distinct voltages. Typical voltages in a TTL-based digital system are, for example, 0 V and 5 V. This means each TTL logic component produces an output of either 0 V or 5 V and responds only to inputs of either 0 V or 5 V. This is fairly simple and straightforward and leads us to consider that a digital device is either off (0 V) or on (5 V). Any two voltages can represent on and off conditions, but certain voltage levels (such as 0 V and 5 V) are standardized by industry. Another common set of standardized voltages used in digital systems is 3.3 V and 0 V.

The logic devices are actually more flexible and variable than mentioned. Using TTL again as an example, the output of a TTL part is considered on if its output voltage is anywhere between 2.4 V and 5 V. We do not care if the output voltage is 2.6 V or 3.7 V, or any other number of volts, for each value between 2.4 V and 5 V is considered to be in the on condition. The off condition exists if the output is anywhere between 0 V and 0.4 V. The voltage ranges make it easier to interconnect parts, since a unique, specific voltage level is not necessary (see Figure 10.4).

The key point is that digital logic devices produce and respond to only two different conditions, on and off. Moreover, because it is easy to build circuits that respond only within two different voltage ranges, the digital parts are inexpensive.

Binary Numbers

Digital system design is based on the binary number system (also called base two), which uses only two numbers, zero and one. This should seem reasonable as digital devices use two voltages. When using the binary number system with digital devices, we say devices that are on are in the on state or high state or one state. The binary number 1 (one) usually signifies the on state. Any device that is off is considered to be in the off state or low state or zero state, and this state is usually designated by the binary number 0 (zero).

A single one or zero is called a bit; a bit is a single binary digit. Using binary numbers and digital devices, designers develop digital systems based on simple on-off responses. In actual practice, the binary numbers used will consist of more than one bit. Eight bits grouped together are common in digital systems. Eight bits make up a byte.

Logic circuit functions are defined by inputs (the wires leading to a circuit) and outputs (the wires coming from a circuit). The signal on an input or output line is always either at the one or zero level. No other condition is possible. Most digital circuits also have multiple inputs (see Figure 10.5). Depending on the one-zero values of these inputs, the circuit output responds with the appropriate one or zero level. The output level is determined by the circuit's logical function. Figure 10.6 shows the electronic symbols used to represent many of the digital devices you will study in this chapter.

Binary numbers can represent many different things in digital systems. A binary number may express a specific numerical value,

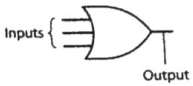
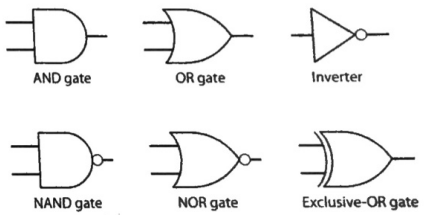


Figure 10.5 | A logic circuit, such as this 3-input OR gate, has signal input wires and a signal output wire.



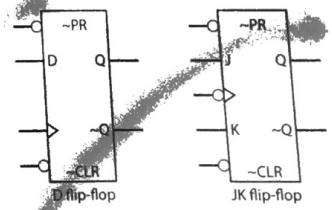


Figure 10.6 | Shown here are some of the digital logic device symbols used in the computer and electronics industry.

or it may convey other information, such as a letter, a color, or a sound level. For example, binary numbers conveniently represent letters and numbers on a computer keyboard, which typically has 104 keys. Each individual key is a simple on-off switch, and all are identical, but the keyboard's function is to produce unique signals for each key. There must be a means to differentiate one key symbol from another.

Every key switch could be wired to the computer, but it is senseless and unnecessary to install one wire per key, or 104 wires total. (If you take a moment to check your computer keyboard connector, you will see that it does not contain 104 pins.) The binary number system makes it easier to represent all the possible key combinations with fewer wires. For example, the letter "A" is represented as eight binary bits (00011110); "B" is represented as a different grouping of eight binary bits (00110000). Each of the 104 keys has its own unique 8-bit binary code (called a scan code), meaning that only eight bits are needed to carry the keyboard information.

Let us examine why a few bits can carry so much information. The secret is to understand the relationship between the number of bits and the combinations of binary numbers the bits produce. We already know that one bit may represent a one or a zero. Based on this fact, two bits will represent exactly four combinations of ones and zeros (00, 01, 10, 11). Taking this concept a little further, you will see that there is always a mathematical relationship between the number of bits and the number of combinations. This relationship is determined with the following formula:

$$\text{Number of Combinations} = 2^N$$

where N = number of bits.

As an example, using four bits we determine the number of combinations:

$$\text{Number of Combinations} = 2^4$$

$$\text{Number of Combinations} = 2 \times 2 \times 2 \times 2$$

$$\text{Number of Combinations} = 16$$

By running a few numbers, we see that each additional bit doubles the number of combinations. Can you figure out how many combinations are possible with 16 bits?

When designing digital circuits, binary numbers are listed in a counting sequence. This makes it easy for the designer to track individual input combinations and the meaning for each in a particular design. We increase a binary count the same way we increase a decimal count, one value at a time. In addition, just as the number of digits determines the largest decimal number one may represent, the number of bits determines the largest binary number one may represent. For example, using three decimal digits, you can count from zero to 999, which represents one thousand unique decimal values. The binary counting sequence using four binary bits is shown in Figure 10.7. Four bits permits us to count from 0000 to 1111, a range which represents sixteen unique binary values.

Binary Numbers	Decimal Numbers
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

ure 10.7 Binary numbers are unted in sequence, just as decimal nbers are.

Each bit in a binary number has a numerical value or weight based on its position in the number. This, again, is similar to the decimal system. For example, the decimal number 234 reads four times one, plus three times ten, plus two times one hundred. The weight of each decimal position increases by ten for every decimal position to the left. In the binary system, the weight *doubles* for every binary position to the left.

For binary numbers, the value of the rightmost bit position is always one. The value of the next position to the left is two. Moving further left, the weights of the positions are 4, 8, 16, 32, 64, 128 and so on. Using this knowledge, we can easily count in a binary sequence and have a method for relating the numerical values of binary and decimal numbers. For example, the decimal value of the four-bit binary number 0110 is found by adding the binary weights of each one-level bit position (we ignore the zero bits). Thus, 1 × 2, plus 1 × 4, equals 6. A binary 0110 is equal to a decimal six.

Important Logic Functions

Just a few basic logic functions form the foundation for most digital designs. These logic functions are defined by the output signal produced for certain combinations of input signals. We discuss these functions next.

AND Gate The AND gate is the digital device producing the AND function. The AND gate produces a one-level output signal when all the gates' inputs are at one levels. When any AND gate input is zero, the output is zero. For example, an AND gate with four input wires may receive any of sixteen possible binary input combinations. Only the combination consisting of all ones produces an output level of one. Each of the fifteen remaining input combinations has at least one of its input wires equal to zero, so each combination produces a zero output level. Typical devices have between two and sixty-four inputs.

Incidentally, the term "gate," when used with logic devices, has a different meaning than the term "gate" as it was used in the last chapter in the discussion of FETs. An FET's gate is a physical part of the FET. In logic devices, the term "gate" refers to the control a logic device has on input signals.

How is something as fundamental as an AND gate used? An example is the familiar CTRL-ALT-Delete key sequence you might type on your computer. An AND function is used to tell the computer that all three keys have been pressed simultaneously. The computer recognizes the sequence only when CTRL, ALT, and Delete are pressed at the same time. All three keys must provide a high-level signal at the same moment for the sequence to be recognized. Figure 10.8 shows a logic schematic illustrating this AND operation.

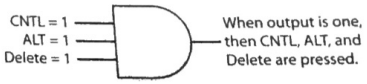


Figure 10.8 | This simple circuit shows how an AND Gate is used to detect the Ctrl-Alt-Delete key sequence.

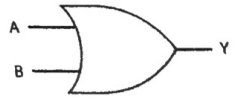
OR and Exclusive-OR Gates The OR gate produces another primary logic function. The OR gate output is one if even one of its inputs is one; the output is zero only when all the inputs are zero. Figure 10.9 shows that the output is one when input A *or* B *or* C is one. You can also see that the OR gate produces an entirely different result than the AND gate.

An important variation of the OR function is the Exclusive-OR (EX-OR) gate, which produces a one output when either of the two inputs is one. When both inputs are ones or both are zeros, the EX-OR produces a zero output. The EX-OR function is used extensively in mathematical, error checking, and encryption circuitry.

Inverter, NAND, NOR Changing a signal's level from one to zero or vice-versa is an important logical operation accomplished by a circuit called an inverter or NOT gate. The inverter has a single input and a single output, and its sole function is to flip the logic level. When inverters are combined with ANDs and ORs, two additional logic functions result, the NAND and NOR.

2-Input OR

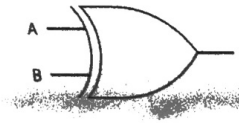
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1



(a)

2-Input Exclusive-OR

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0



(b)

Figure 10.9 | This illustration shows the symbol and truth table for a 2-input OR gate (a), and the symbol and truth table for an Exclusive-OR gate (b).

3-Input NAND Gate

A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

(a)

3-Input NOR Gate

A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

(b)

Figure 10.10 | This illustration shows how the symbols and truth tables are represented for a 3-Input NAND gate (a) and a 3-Input NOR gate (b).

NAND stands for 'Not AND' and NOR stands for 'Not OR.' These devices produce the inverted responses of the AND and OR gates as shown in Figure 10.10. NANDs and NORs are powerful logic functions. It is possible to design and build an entire digital system just from NAND or NOR devices (lots of them, of course). You may even own devices built with these circuits, such as MP3 players and USB thumb drives, which use flash memory chips based on NAND technology.

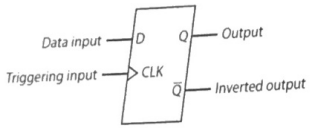


Figure 10.11 | A D-Flip-Flop has inputs labeled D and clock (CLK), as well as two complementary outputs labeled Q and not Q.

Flip-Flops The ability to store ones and zeros is also important in a digital system design. Memory circuits do this. Digital systems need memory to store any quantity of information, from a single bit to many gigabits.

Flip-flops are digital circuits that store a single bit (see Figure 10.11 for an example of a flip-flop). They respond to special timing or triggering signals that indicate exactly when a bit is ready to be stored (for instance, when numbers change on a digital clock). The flip-flop dutifully retains the stored value until another trigger signal causes a change to occur.

Small numbers of flip-flops combined together are called registers. Registers are common storage areas in computers and typically store between 8 and 64 bits. Memory devices, on the other hand, are built to store many millions of bytes of information at a time. Memory chips can be thought of as large numbers of flip-flops organized to work together to efficiently store massive amounts of binary information. Memory devices are discussed in detail in another section of this chapter.

Combinatorial and Sequential Circuits

Digital circuits are classified by their operating characteristics. There are two general classes of digital circuitry—combinatorial and sequential circuits.

In combinatorial circuits, the output is directly dependent on circuit input signals. Sequential circuitry contains storage elements so that the output level is dependent on existing input signals as well as previously stored data. Sequential circuits store data in flip-flops or memory.

In Figure 10.12, a truth table shows every binary combination possible for the number of inputs in a combinatorial circuit. Truth tables are created to detail how the circuit should react for every input combination. In this example, A, B, C, and D are the names assigned to the circuit inputs. The output is identified with the name Y. Since there are four inputs, this circuit has $2^4 = 16$ input combinations. Notice how the input combinations are listed in an ascending binary count. Each binary number represents a possible set of signals for the circuit. Here, the truth table describes how the circuit reacts for each set of input values.

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

This truth table shows that output Y will produce a one level only when combinations ABCD = 0110 or ABCD = 1100 are present at the circuit's inputs. All other input combinations are designated to produce a zero output response. AND, OR, and Invert gates are used, as shown in Figure 10.12, to create this circuit. After the truth table is created to describe the circuit response, AND gates and inverters are used to decode the combinations that will produce the one-level output. An OR gate further combines the two decoded signals into one final output. The resulting circuit produces a one output only for two specific input combinations. So, by default, the same circuit produces a zero level output for all remaining input combinations.

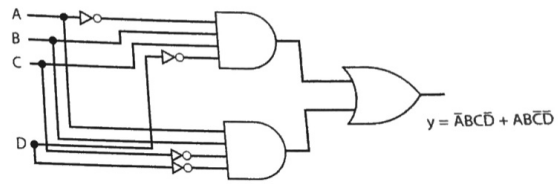


Figure 10.12 | This combinatorial circuit is designed to produce the outputs levels shown in the truth table for each corresponding input combination.

The example in Figure 10.12 shows the most common type of combinatorial circuit structure. The circuit is comprised of AND gates connected to an OR gate. Circuits with this structure are called AND-OR networks.

Engineers use many techniques to design combinatorial circuits. It is possible to design two circuits that produce the same output, but one will require fewer parts, so digital designers employ specialized minimization techniques to reduce circuitry size without changing its function. For this purpose, computer aided design is a helpful tool.

One common technique in designing and simplifying combinatorial circuitry is Karnaugh Mapping (K-map). K-maps graphically rearrange truth table combinations to make circuit minimization obvious to the trained eye. This is important because circuits with fewer parts are smaller, more energy efficient, and more reliable.

Sequential circuit design is much more involved than combinatorial design and requires specialized design knowledge. The counting circuit described in the next section is a simple example of a sequential circuit. The microprocessor chip, discussed later in this chapter, is an example of a highly complex sequential system.

As you have seen, basic digital functions combine to create complex circuits. Complex circuits, in turn, combine to produce computers and other sophisticated digital systems. Computer hardware is composed of key architectural elements, including registers, counters, and ALUs. Within these systems, certain circuits tend to recur. We will examine these important commonly used circuits next.

Counters

Counters are digital circuits that react to input signals and consequently produce a binary counting sequence. Some counters are designed as up counters, which produce a binary counting sequence from zero to some final maximum value.

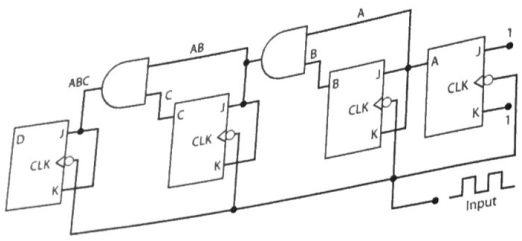


Figure 10.13 | A, B, C, and D are the four outputs of this 4-bit Binary Up Counter.

Down counters provide a count sequence, which decreases from a maximum value to zero. A common microwave oven uses a down counter. For example, after cooking time is set on the microwave, the oven's digital display, controlled by a down counter, counts down to zero. The counter's zero value visually informs the user that cooking time has ended. The counter may also generate other signals that turn off the heat, ring a bell, or flash a message.

Counters use flip-flops in their designs. Figure 10.13 illustrates the circuitry for a 4-bit up counter. 4-bit means that four flip-flops are used to build the counter and each provides an output signal. The number of flip-flops used determines the counting range. The counter in Figure 10.13 counts from 0000 to 1111, which comprises sixteen different counts. We refer to individual counts as states. In digital systems, each counter state may drive other circuitry in the system. Counter circuits are particularly useful for controlling the sequencing of events. In the microwave oven example, the zero state of the counter signals additional circuitry to shut down the oven.

So far, we have not defined what the counter is actually counting. We often refer to the counter's input signal as a clock pulse. The actual clocking signal may be a periodic electronic timing signal or a signal generated by a sensor. When a clock pulse arrives at the counter's input, the counter advances its count by one state. For example, if the counter is currently in its fifth state, a clock pulse will advance the count to the sixth state.

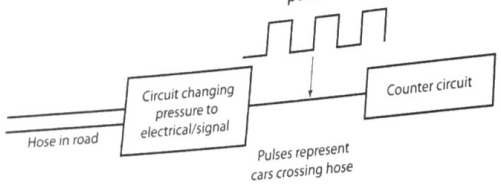
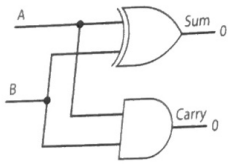


Figure 10.14 | A clocking signal derived from pressure changes in a hose can trigger a counter and thus be used to count automobile traffic.

Here is a simple example of actual event counting. Have you ever driven over a hose placed across a road by the highway department? The hose is an input device used to help count the number of cars using a highway. A lack of pressure in the hose might produce a digital zero while an increase in pressure could produce a digital one level. The weight of your car compresses the hose as you drive over it, changing the air pressure within the hose. A pressure sensor detects the air pressure change and converts the change into an electrical signal. These changing signals become clocking signals for a counter. In this arrangement, the counter counts cars as the changes in pressure are detected (see Figure 10.14).

Arithmetic and Logic Unit

A critical task for any computer is performing mathematical operations. The digital circuitry responsible for mathematics is the Arithmetic and Logic Unit, or ALU. The ALU circuitry usually performs mathematical operations such as adding, subtracting, multiplying, and dividing. Complex ALUs perform high-level mathematical operations, as well. Logical operations such as AND, OR, and Invert are also implemented within the ALU. Mathematical circuitry uses many logic gates. In most computer processing chips, a significant amount of the circuitry is devoted to mathematical operations.



A	B	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

re 10.15 | This half adder circuit ires only two gates to add two bit numbers together.

We can demonstrate the use of basic logic gates in mathematical circuitry by examining the simplest addition circuit, the half adder. Figure 10.15 shows this circuit, along with its truth table, which shows all four possible additions. The half

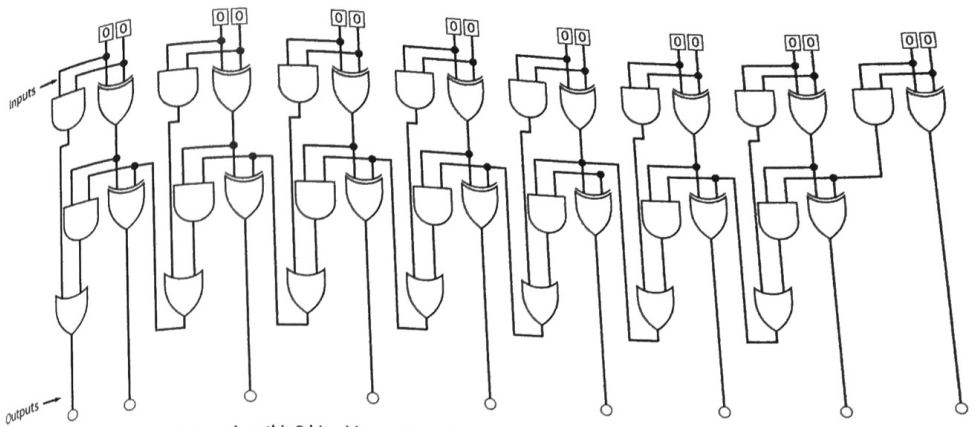


Figure 10.16 | Large scale adders, such as this 8-bit adder, are formed by cascading many half and full adders.

adder adds two individual bits together. A two-bit answer is produced on the half adder outputs.

The half adder circuit must have two outputs for results because the largest result in binary requires two places (1 + 1 = 10). These outputs are named sum, which is the least significant bit of the result, and carry, which is the most significant bit of the result. The sum is generated using an Exclusive-OR gate, while the carry is generated using an AND gate.

Figure 10.16 shows the circuitry for a larger but relatively simple 8-bit adder (for example, 10000111 + 10000100 = 100001011). An 8-bit adder is capable of adding any two 8-bit numbers together. Notice how the number of logic gates needed for this level of addition has increased dramatically. Because logic gates work at electronic speeds, this adder easily performs over 1.5 million additions per second. Nonetheless, in modern computing systems, this is slow. So, for faster computation, even more circuitry is required, adding substantially to the amount of the overall ALU circuitry. You can imagine how many gates would be necessary to add two 64-bit numbers together at high speeds, which is commonplace in desktop microprocessor chips. Also, keep in mind that we are examining only adder circuits. A full ALU will have considerable additional circuitry to support all the other mathematical and logical functions required.

SECTION ONE FEEDBACK >

1. Calculate the decimal value of the binary number 10101.
2. Sketch the truth table for a 5-input OR gate.
3. Calculate the number of input combinations a 64-input AND gate will have. How many of these combinations produce a one-level output?
4. How could a counter circuit be used to count paint cans moving along a conveyor line in a factory? How could the counter help group eight cans together for packaging?
5. Prove that the 8-bit adder circuitry shown in Figure 10.16 adds 10010111 + 11001010 and produces an answer of 101100001.

SECTION 2: Memory

KEY IDEAS >

- Memory is necessary in all computer systems.
- Many different memory technologies exist.
- Memory chips contain electrical circuitry capable of storing a one or a zero.
- A write operation places information into a memory chip; a read operation retrieves previously stored information from memory.
- Volatility describes a memory chip's ability to retain information once it is stored.

Computer systems cannot function without memory devices. Many different memory technologies exist, each possessing characteristics desirable for specific applications. Typical computer systems use several of these memory technologies.

Data in Memory Chips

Memory chips contain electrical circuitry capable of storing a one or a zero. Stored information is called data. Electrical signals activated memory's storage circuitry to accept digital values as data and then store the data at specific locations inside the memory. This storage capability of a memory device is similar to that of a flip-flop. The difference between the flip-flop and memory is that a flip-flop stores one bit of information, whereas a memory chip stores millions of bits of data (see Figure 10.17). This is why the computer industry expresses memory sizes in Megabytes (MB) (approximately a million bytes) and Gigabytes (GB) (approximately a billion bytes). Integrated circuit fabrication makes it possible for a single memory chip to hold vast amounts of data. Memory is necessary in all computer systems.

The enormous storage capacity of a memory chip requires mechanisms to track and retrieve stored information, similar to locating papers in a file cabinet. To understand how memory works, we will discuss the key memory concepts of organization, reading, writing, and addressing.

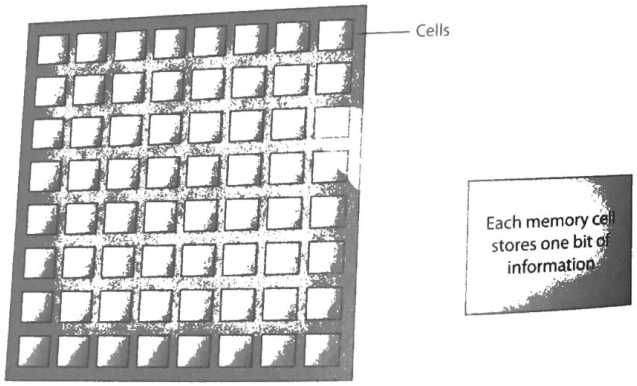


Figure 10.17 | Individual memory chips are made of millions of storage cells, and each cell may store a one or a zero.

Memory Cell

Memory chips store one bit of information within an electrical circuit called a cell. A memory chip has millions of cells, but a typical memory operation (such as storing or retrieving) requires using only a few cells at a time. It is common, for instance, to store or retrieve eight bits at a time. The cell construction varies from technology to technology. Some cell structures are flip-flops, including that of an important type of memory called static RAM (SRAM). The cells in another memory technology, called dynamic RAM (DRAM), are constructed from one transistor and one capacitor. There are many other technologies available. Each has interesting operating characteristics, but the key work of any cell is to store and retain a single binary bit.

As mentioned, a useful memory chip has millions of cells, and keeping track of the cells is important. Memory chips are manufactured with cells organized in rows and columns. This organizational pattern also describes the functional specification of the memory chip. For example, a chip that stores or retrieves eight bits at one time and has 16 million groups of eight cells is called a 16M × 8 memory chip. The 16M represents the chip's 16 million locations or rows, and the 8 represents eight columns, which is the number of bits of information available at one time. In this example, the eight bits of data are the word size, or amount of data used at any one time. Once you understand memory organization, the naming convention of chips is easy to interpret. A memory chip must have the circuitry to select any one of its row locations at any time. Addressing circuitry, discussed in more detail shortly, accomplishes this task. Addresses are the numerical way to locate information in a memory chip.

What Is a Million?

▷ In the computer memory business, one million is an approximate amount. With the binary number system, the closest value to one million is 1,048,576 because 2^{20} equals this number. It is more convenient to say "one million" than "1,048,576."

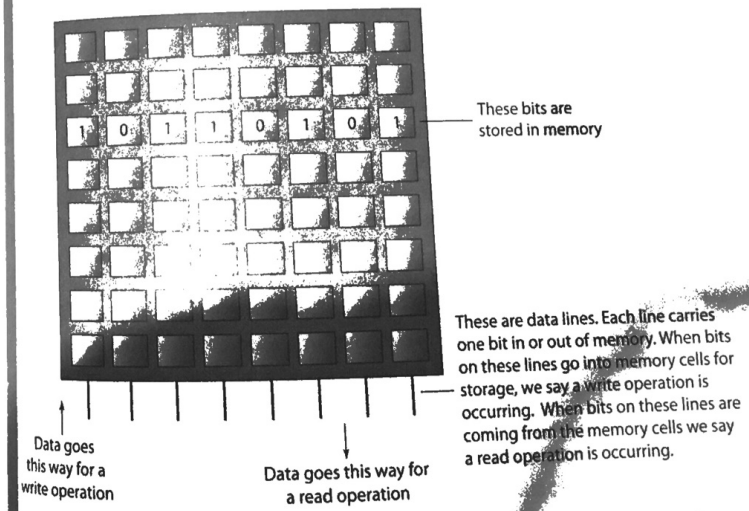


Figure 10.18 | Data flows into a memory chip during a write operation and flows out of the memory chip during a read operation.

Writing and Reading

In memory systems, Writing is the act of storing information, and reading is the act of retrieving it (see Figure 10.18). There are pins on a memory chip exclusively for data. For instance, an 8-bit memory chip has eight data pins, which connect external data to the memory chip. Electrical signals are sent to the memory chip control pins, commanding the chip to store information in selected or addressed cells. The timing of signals ensures that the write operation occurs at the correct moment. The designer is responsible for choreographing the relationship between all signal and data lines.

Eventually, a read operation retrieves previously stored information from memory. Electrical signals command the chip to copy information from addressed memory cells and place it onto data pins for another device, such as the CPU, to use. The read operation is the inverse of the write operation (see Figure 10.19).

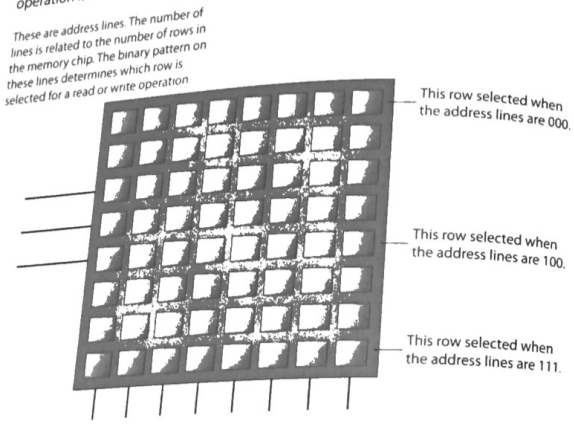


Figure 10.19 | Each memory operation involves only a few cells, which are selected by the addressing circuitry of the chip.

Memory Addressing

Let us examine in more detail how addressing circuitry controls the cells used in read and write operations. A certain number of cells is used for any one operation, and the number of cells used is based on the way the chip is organized. For example, a certain chip manufactured to read or write eight bits at a time reads or writes data in one byte quantities, and each one byte storage location has a unique address. A memory chip capable of storing a million bytes is organized as a 1M × 8 chip. This means there are approximately one million 8-bit locations or addresses on the chip. Figure 10.20 lists common memory chip sizes according to their address and word size organization.

Previously, we mentioned that electrical control signals are necessary in commanding a memory chip to perform a read or write operation. Memory chips have numerous pins for these signals, and many carry the chip's addressing information. As a simple example, let us say that a chip has a mere 512 addresses. We would view the addresses as a sequence of unique numbers ranging from 0 to 511. Each unique address represents or points to a specific group of cells in which data are stored. Therefore, in order to activate any specific group of cells for reading or writing, the specific cell address must be supplied to the chip. If the chip stores eight bits per address, then each address points to eight specific cells.

Addresses are binary numbers, and there is a relationship between the number of addresses on a memory chip and the number of pins needed to represent those addresses. Recall that in the binary system, $2^9 = 512$. This tells us that the memory chip requires nine pins just for the addressing information. Since we keep track of addresses in a binary counting order, the first address in this example is 000000000, while the last is 111111111. The 9-bit combinations from 000000000 to 111111111 represent all 512 addresses.

It is helpful to visualize memory addresses as house addresses. Within the memory chip, an electrical version of mail delivery takes place. In the same way your mail carrier reads and decodes the address on an envelope and delivers it correctly, addressing circuits decode each address supplied on the memory chip's address lines and deliver a signal, activating the appropriate cells.

Memory Characteristics

Modern computing systems use several memory technologies, since no single technology meets all the requirements of a computer system. The key technology considerations in selecting a memory chip for design are speed, density, and volatility.

Speed The speed of a memory chip determines how quickly a read or write operation occurs. Computer chips carry out read and write operations in nanoseconds. We define the overall speed of memory chips by their access time, the time it takes for data to appear during a read operation or the time it takes to store data during a write operation. (A write operation places information into a memory chip.) Ideally, memory chips should work as fast as any of the other components of a computer system. Unfortunately, this is currently not technically or economically possible. Therefore, designers use various memory technologies in computer system designs.

Density Density refers to the number of cells per chip. The cell's electrical structure determines its physical size, which, in turn, determines the number of cells possible on a chip: The more cells, the greater the density of the chip. Building computer systems with fewer chips means using fewer parts, thus keeping costs down.

Volatility Volatility describes a memory chip's ability to retain information once it is stored. Your home computer, for example, has both volatile and nonvolatile memory chips. The nonvolatile chips retain their data whether the machine is on or off. The volatile chips lose information as soon as the power is cut.

Take a closer look now at how these three characteristics affect a home computer system.

Unfortunately, no single memory technology provides high speed, great density, and nonvolatility all in one chip. Computer designers use the memory technologies best suited for specific computer functions.

Consider the process of turning on your computer. When power is first applied, we say the system boots. This means that the computer reads simple instructions from a memory chip in order to prepare the computer for use. Initialization programs, which test the machine's components and start some of the computer's hardware, are stored in nonvolatile memory. Therefore, every time the system starts, the same instructions run first, readying the machine for your use.

Say you start a video game. The game program is loaded from a hard drive into dense memory chips so that the millions of complex instructions defining the game are ready for the computer to execute. You surely want the game to run fast, so some of the game instructions are stored in high-speed memory chips in order to keep up with the computer's speedy processor chip. The three memory characteristics of speed, density, and volatility have all affected the operation of a computer system.

Total Capacity	Organization
256k	32k × 8
1M	128k × 8
2M	128k × 16
2M	256k × 8
4M	256k × 16
4M	512k × 8
8M	512k × 16
16M	512k × 36
4M	1M × 4
8M	1M × 8
16M	1M × 16
36M	1M × 36
16M	2M × 8
32M	2M × 16
4M	4M × 1
32M	4M × 8
64M	4M × 16
128M	4M × 36
64M	8M × 8
128M	8M × 16
128M	16M × 8

Figure 10.20 | Memory chip sizes are organized by address range and word size.

RAM and ROM

Semiconductor memory technologies fall into two main classifications, Random Access Memory (RAM) and Read Only Memory (ROM) (see Figure 10.21 for a more in depth classification of memory technologies). RAM devices, which make up the main memory in a computer system, tend to be volatile. RAM chips directly connect to microprocessor chips because RAM can keep up with the microprocessor's operating speed; RAM's best asset is that read and write operations occur equally fast. ROM chips have unequal read and write access times, so ROM technologies are not practical as computer main memory. The significant advantage of ROM technologies over RAM is nonvolatility. Once data are stored on a ROM chip, it stays there with or without electrical power. ROMs do need power to read data, but do not lose data without power. This allows you to turn off your computer knowing it will work when you turn it on again.

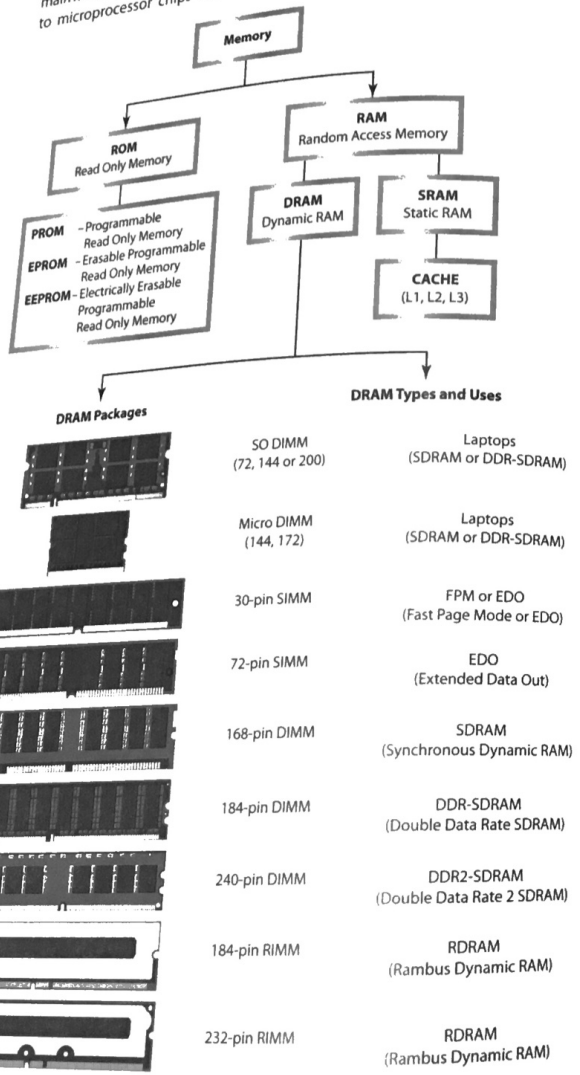


Figure 10.21 | Many specific memory technologies exist. Each fits into the general category of RAM or ROM memory.

SRAM

Static RAM (SRAM) memory chips are among the fastest RAM memory technologies available (Figure 10.22), so SRAM is the memory technology most often connected to the microprocessor. In fact, the powerful microprocessor chips used in home computers and workstations frequently have SRAM memory fabricated right on the microprocessor chip. Microprocessors execute instructions very quickly with this architecture, since the instructions are stored in the SRAM right next to the processor. In the field of computer engineering, cache memory is the name given to high speed SRAM used this way.

The main disadvantage of SRAM is its relatively low density compared to other technologies. Each SRAM storage cell requires a considerable amount of circuitry to function, and each cell takes up a fair amount of space. Consequently, a chip will hold fewer cells. This lack of storage capacity makes SRAM expensive for large memory systems.

Figure 10.22 | Static RAM (SRAM) chips are noted for their fast access speeds.

DRAM

Dynamic Random Access Memory (DRAM) chips are used when large amounts of memory are needed (Figure 10.23). DRAM chips are dense because each storage cell consists of only one transistor and one capacitor. This is a small-sized cell, making it possible to pack millions of cells on a single chip. If you buy a home computer, you will find it important to consider the amount of DRAM memory you buy.

DRAM chips have the peculiar characteristic of losing information after a short time, even with power applied to the chip. If a one value is stored in a DRAM cell, for instance, the information lasts for two to four thousandths of a second and then is gone. The short storage time occurs because of the way DRAM stores data. Ones are stored as an electric charge in microscopic capacitors. Although it is easy to charge the capacitor to a level representing the one value, the small capacitor size prevents much charge from being stored. Furthermore, the capacitors' poor retention characteristic means that charge will "leak off" after a short amount of time. In spite of its short storage time, the DRAM chip is still useful.

Figure 10.23 | Dynamic RAM (DRAM) chips are noted for their great storage capacity.

DRAM refresh circuits control DRAM memory systems to counteract the leakage problems. Using refresh circuits, data are periodically read and rewritten so information is retained as long as each cell is periodically used. The refresh circuitry is complex and makes DRAM memory designs complex. Even with this limitation, the low cost of DRAM makes it the best choice for large memory system applications.

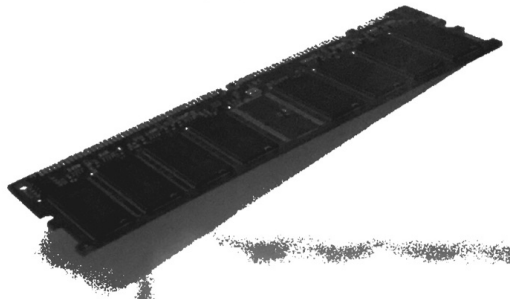


Figure 10.24 | Here, eight DRAM chips are connected to operate together on a module.

DRAMs are typically connected together on small circuit cards called DIMMs (Dual In-line Memory Modules) as shown in Figure 10.24. A DIMM plugs into a connector slot in a computer system so that the user can control the amount of memory.

EEPROM

Electrically Erasable Programmable Read Only Memory (EEPROM) is a ROM technology. An EEPROM stores information in a special cell structure known as

a floating gate. The floating gate is made from semiconductor material and may hold electrons, much like a capacitor. It is either charged or uncharged, yielding the two distinct binary levels. Unlike the capacitor used in a DRAM memory chip, the floating gate will not lose any charge and so is nonvolatile.

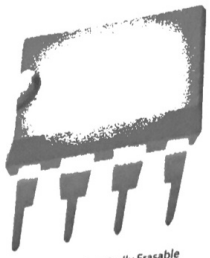


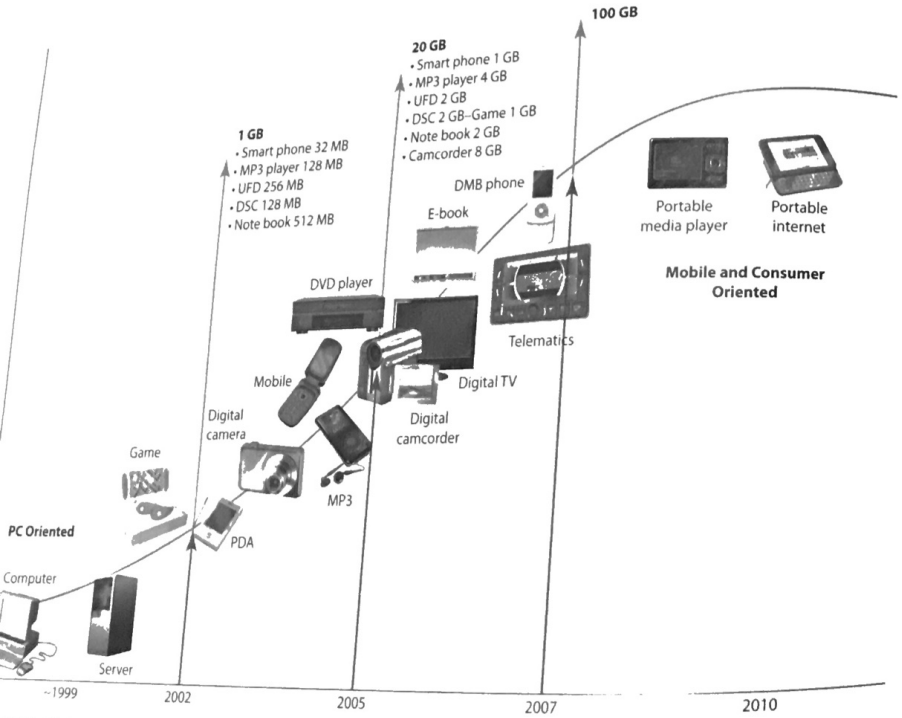
Figure 10.25 The Electrically Erasable Programmable ROM (EEPROM) chip can hold data even without power applied.

EEPROMs require elevated voltages for write operations in order to force charge into the floating gate, and they also require specialized equipment called device programmers to store data. Once programmed with data, the EEPROM chip is installed in a computer system (see Figure 10.25). If the information on the EEPROM needs updating, the chip must be removed from the computer system and returned to a device programmer for modification. This is called erasing the chip. Obviously, this process is very time consuming, so designers often build the device-programming circuitry into the computer. However, the write operation, whether done in-system or out-of-system, is slow. Additionally, the device programming circuitry is costly.

The EEPROM is electrically erasable. This key feature differentiates the EEPROM from the older EPROM (Erasable Programmable Read Only Memory), which can be erased only when out-of-circuit and by exposure to ultraviolet light. Only after this slow erasing process is the old EPROM chip reprogrammable.

FLASH Memory

Flash memory technology is similar to that of EEPROM and currently dominates the nonvolatile memory market. Flash stores data in many digital products, some of which are shown in Figure 10.26. Digital cameras use flash chips to store images;



10.26 Flash memory has operational characteristics similar to that of EEPROM and is the memory technology currently dominating the atile memory market.

MP3 players use flash to store music and video files; your USB thumb drive uses flash to store your term paper. In complex equipment, flash stores programming and operating system instructions. As in the EEPROM, flash write time is slower than read time. Unlike EEPROM, in which one can erase a single byte at a time, flash can be erased only in large blocks of information.

Flash is dense and therefore useful for applications in which large amounts of storage are necessary and slower write speeds tolerable. In an MP3 player, for example, a delay is acceptable when downloading a music file. In listening to a song, however, we expect no delays and appreciate flash's fast read operation.

MRAM

Magnetoresistive Random Access Memory (MRAM) is a promising nonvolatile memory technology. MRAM stores information magnetically, whereas the other ROM technologies store it electrically. MRAM read and write cycles are fast, which overcomes one of the serious limitations of other nonvolatile technologies. Also, MRAM densities are very high, giving this technology the potential to be the perfect memory chip. MRAM is still in development.

ENGINEERING QUICK TAKE

Increasing Memory System Capacity

Computer systems require large amounts of memory. Usually, one memory chip is insufficient to meet all requirements, so designers interconnect individual memory chips to create a larger memory system.

Memory capacity may be increased in two ways. If the designer's goal is to read or write more information at one time, she has to increase the data or word size of the memory system. If the goal is to increase the number of storage locations, she must increase the address range. Of course, she can also increase both dimensions if needed.

Let us assume we are designing a small computer needing a 512k × 8 memory system. The memory system design will use as many 256k × 1 memory chips as necessary because we have determined that these are readily available and inexpensive. We also know that a memory chip with 256k addresses will have 18 address lines (2^{18} = 256k). Since we are expanding the memory to 512k, we will need more addressing lines. But, how many will we need? Based on the fact that each additional bit doubles the number of combinations, we use one more bit, changing 18 to 19, and calculate that 2^{19} = 512k. We have now determined that the specified 512k addresses require a total of 19 address lines in order to access every storage location in the system. We also notice that the specifications call for eight data lines.

Our next calculation will show the number of 256k × 1 chips needed for this system. When we multiply 512k × 8, we get 4,096k. This informs us that the memory system has 4,096k total cells.

We determine the number of cells in each chip by multiplying 256k × 1 and getting 256k cells per chip.

We next divide the total number of cells required by the number of cells per chip (4,096k / 256k). This tells us that the system needs sixteen chips (4,096k / 256k = 16).

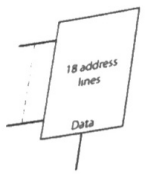


Figure 10.27 It requires 18 address lines to fully address a 256k memory chip.

Next, we determine how the chips should be interconnected. Figure 10.27 is a representation of the address and data lines on a single 256k × 1 chip. Since the chip has 256k addresses, 18 address lines are present per chip. We know that the overall memory system requires 19 total address lines. It looks as if we have far more address lines than we can possibly use since sixteen chips with 18 address lines each equals a total of 288 address lines. We will need to solve this dilemma.

There is also a single data line on each of the sixteen chips, and we need to provide eight total data lines for the system. Once again, we seem to have more lines than we need.

Figure 10.28 shows how we will build the memory system. Of the sixteen chips, eight of them will connect their data lines to the other eight. As two lines connect, they fuse into one data line. Eight individual data lines result.

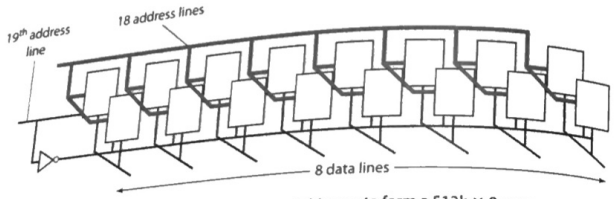


Figure 10.28 Sixteen 256k × 1 chips are connected this way to form a 512k × 8 memory system.

We connect one address line from the first chip to the corresponding address lines on all the other chips. As we do this for each line on every chip, we combine lines to create eighteen total address lines. You can see that all the individual chips' lines are simply connected in parallel.

However, the system requires one additional address line to bring the final system total to nineteen. Each memory chip has a chip-select input line, which we can use as an address line. This nineteenth address line is the most significant one because it determines which half of the sixteen chips operates together at any one time. If all sixteen chips were active simultaneously, data from one-half of the system would electrically conflict with data from the other half, so it is critically important in this design that only eight chips are active at once. When chip-select is active, the chip places data on its data lines; otherwise, the lines are electrically deactivated.

SECTION TWO FEEDBACK >

1. What memory technology is most useful for [illegible] A computer system?
2. How many storage cells are in a 4M × 8 flash [illegible]
3. What is the word size for a 4M × 8 flash ch[illegible] many address lines does this chip have?

KEY IDEAS >

- Computer systems use microprocessor chips as the main computing device.
- The heart of any microprocessor chip is the central processing unit, or CPU.
- The main function of the CPU is to process instructions.
- Processing each instruction occurs in two phases called fetch and execute.
- Instructions are binary commands directing the CPU to carry out relatively simple actions.

What do you think of when you hear the word "computer"? Do you visualize the desktop machines in your computer lab at school? Maybe the wireless notebook your friend uses to check e-mail comes to mind. Possibly, you are thinking about the device controlling the airbag in your car. It is possible that you know your cell phone has a computer chip controlling how you download, organize, and listen to music.

Figure 10.29 | Embedded processors are computer chips placed inside a product in order to control the product's functions.

Microcontrollers and Microprocessors

Familiar computers, such as desktops and notebooks, are obvious to most people. Other computers are not so obvious. Many common products are computer controlled, but do not appear computer-like to the user. Computer chips hidden inside a product (joystick, coffeemaker, windshield wipers) are embedded processors (see Figure 10.29), sometimes known as microcontrollers. Sales of embedded processors exceed those of any other computing chips. Usually, an embedded processor is programmed to do very specific things, and the user of the product is unaware of its existence. For instance, when sending a text message, are you thinking, "I'm using a computer"? Probably not, because you are concentrating on the keys you press to compose the message. An embedded processor is working behind the scenes to handle the many technical processes required to compose and send the message. Of course, desktop and notebook computers also need computing chips to function. In these general-purpose computers, the digital hardware doing the main computing is the microprocessor. Computer systems use microprocessor chips as the main computing device.

The modern microprocessor is an integrated circuit containing millions of transistors that form the digital logic architecture of the chip. The prefix "micro" distinguishes these chips from older, larger computers. When integrated circuit technology advanced to the point where millions of transistors per chip became feasible, the microprocessor era began, and the computer on a chip became reality. The Intel Pentium and the AMD Opteron lines of chips are examples of microprocessors.

The next section defines more specifically the key hardware found inside a modern computer microprocessor chip.

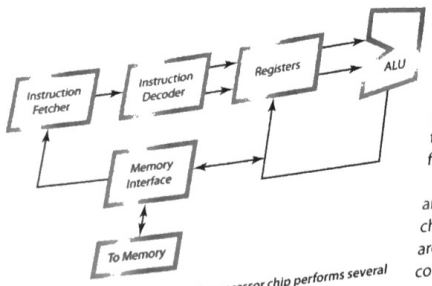


Figure 10.30 The central processor unit of a processor chip performs several key functions, as detailed in this block diagram.

The CPU

The heart of any processor chip is the central processing unit, or CPU (see Figure 10.30). The CPU circuitry performs computing functions at high speeds. These computing functions may include a simple process, such as moving a byte of information from one part of the CPU to another, or a more complicated computing function, such as multiplying two 64-bit numbers.

The structure of logic circuitry in a CPU defines its architecture. Computer architecture for a Pentium chip in a desktop computer is different from the architecture of an embedded processor used to control a microwave oven, just as the architecture of a castle is different from that of a log home.

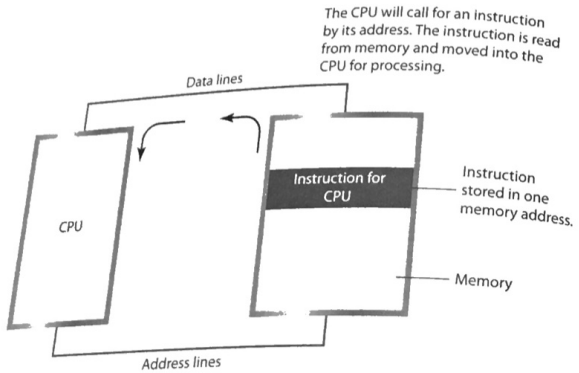


Figure 10.31 | **CPUs begin to execute instructions after first reading the instruction code from memory.**

ruction ode	Instruction Binary Pattern
A, R0	11101000
R6	00101110
	00000011

hese examples of basic nstructions show how ns of ones and zeros CPU operations.

Computer Instructions

The main function of the CPU is to process instructions. Instructions are binary commands directing the CPU to carry out relatively simple actions. Each specific binary pattern of the microprocessor defines a small task. When many instructions occur in sequence, we say the CPU is executing a program. A program running on a computer defines what the computer can do at that moment. The CPU architecture determines whether the instructions are simple or complicated.

Instructions are very specific patterns of ones and zeros, and each pattern triggers the CPU to perform a specific operation. Every microprocessor has an instruction set, which is the total of all its commands. A typical instruction set may have several hundred instructions.

When a computer runs a program, the program's instructions are stored in memory and called by the CPU one after another for execution (see Figure 10.31). A computer designer, programmer, or computer engineer creates the sequence of instructions executed (see Figure 10.32). The person or team responsible for making the computer work at this level, called the hardware level, must have an in-depth understanding of CPU architecture. Binary instructions, or machine-level instructions, are part of the hardware. They are different from instructions used in

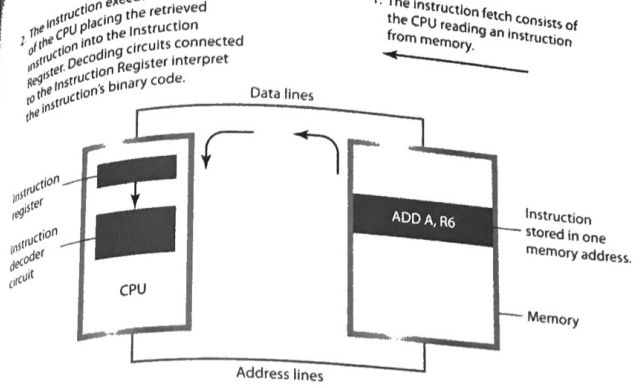


Figure 10.33 | **The overall pattern of CPU operation is to fetch and then execute an instruction.**

high-level programming languages such as Visual Basic, C++, or Java. Programs written using a high-level programming language are converted by compilers into appropriate sequences of machine-level instruction. It is machine-level instructions that actually execute within a microprocessor (see Figure 10.33).

Digital Signal Processor (DSP)

▷ Typical microprocessor chips all have basic operational characteristics, such as the ability to move information from register to register. Sometimes, microprocessor designers customize parts of a chip's architecture to optimize the chip's performance. A **Digital Signal Processor (DSP)** is such a microprocessor. Instructions for DSPs are designed around a logic architecture specifically created to enhance the mathematical capabilities of the chip.

DSPs are used in products that require continuous computer processing. For example, audio filtering in hearing aids eliminates extraneous sound and noise information that the user of the hearing aid would find objectionable. A DSP can be programmed to do this filtering. In this application, the DSP constantly receives audio information as an input and uses complex mathematical operations to examine the audio and sort out the useful and non-useful sound information. A DSP excels at such a task compared to a standard microprocessor because of the detailed high-level mathematical functions built into the chip.

Instruction Execution

Let us examine a microprocessor whose instruction set consists of instructions made from just eight binary bits. This machine could have up to 256 individual instructions. We will examine what happens as the CPU processes these instructions.

In general, processing each instruction occurs in two phases called fetch and execute. The CPU fetches, or obtains, the first instruction from memory by reading