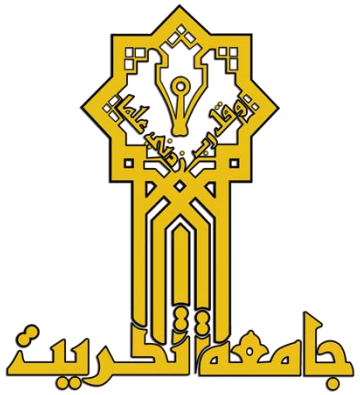


DIGITAL DEVICES AND LOGIC FAMILIES

ELECTRICAL DEPARTMENT

FOURTH STAGE

INSTRUCTOR: DR. ANAS MQDAD

LECTURE FOUR

DATA STORAGE

OUTLINES

Introduction

In previous lecturers we have covered shift registers, which are a type of storage device. The memory devices covered in this chapter are generally used for longer-term storage of larger amounts of data than registers can provide. Computers and other types of systems require the permanent or semipermanent storage of large amounts of binary data. Microprocessor-based systems rely on storage devices for their operation because of the necessity for storing programs and for retaining data during processing. In this chapter semiconductor memories and Magnetic and optical storage media are covered. Also, memory hierarchy and cloud storage are discussed.

Semiconductor Memory Basics

Memory is the portion of a computer or other system that stores binary data. In a computer, memory is accessed millions of times per second, so the requirement for speed and accuracy is paramount. Very fast semiconductor memory is available today in modules with several GB (a gigabyte is one billion bytes) of capacity. These large-memory modules use exactly the same operating principles as smaller units

Units of Binary Data: Bits, Bytes, Nibbles, and Words

As a rule, memories store data in units that have from one to eight bits. The smallest unit of binary data, as you know, is the **bit**. In many applications, data are handled in an 8-bit unit called a **byte** or in multiples of 8-bit units. The byte can be split into two 4-bit units that are called **nibbles**. Bytes can also be grouped into words. The term **word** can have two meanings in computer terminology. In memories, it is defined as a group of bits or bytes that acts as a single entity that can be stored in one memory location. In assembly language, a word is specifically defined as two bytes.

The Basic Memory Array

Each storage element in a memory can retain either a 1 or a 0 and is called a **cell**. Memoriesn are made up of arrays of cells, as illustrated in Figure 1 using 64 cells as an example. Each block in the **memory array** represents one storage cell, and its location can be identified by specifying a row and a column.

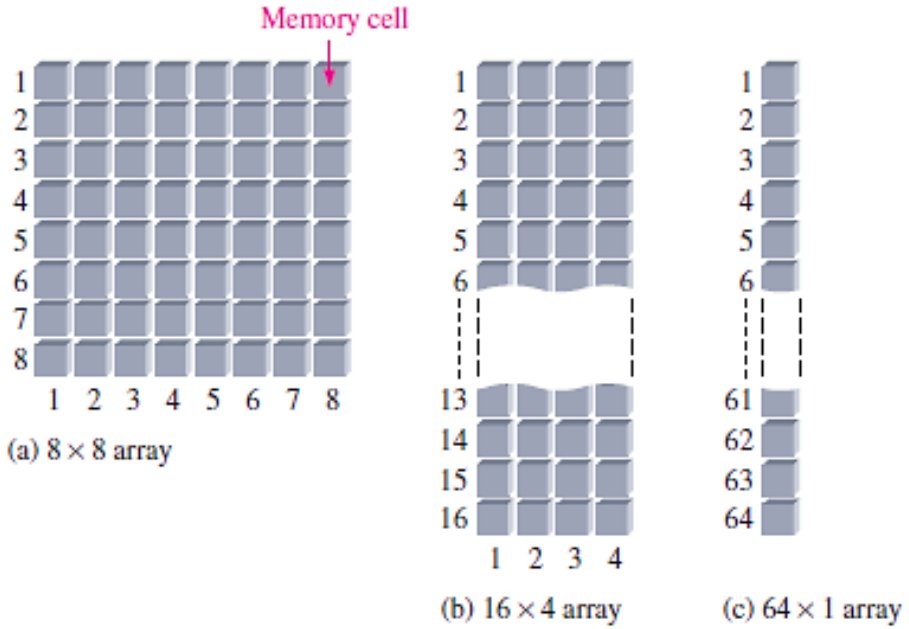
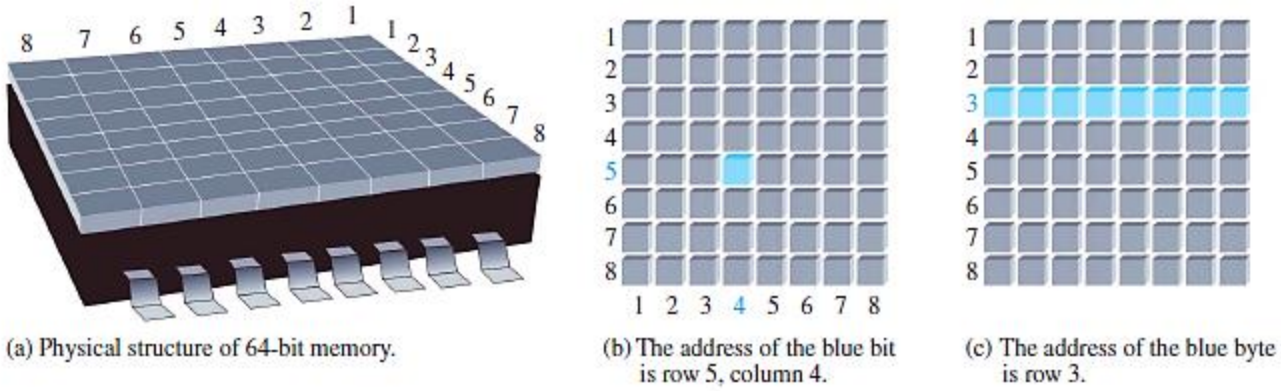


FIGURE 1 A 64-cell memory array organized in three different ways.

The 64-cell array can be organized in several ways based on units of data. Figure 1(a) shows an 8 * 8 array, which can be viewed as either a 64-bit memory or an 8-byte memory. Part (b) shows a 16 * 4 array, which is a 16-nibble memory, and part (c) shows a 64 * 1 array, which is a 64-bit memory. A memory is identified by the number of words it can store times the word size. For example, a 16k * 8 memory can store 16,384 words of eight bits each. The inconsistency here is common in memory terminology. The actual number of words is always a power of 2, which, in this case, is 2^{14} = 16,384. However, it is common practice to state the number to the nearest thousand, in this case, 16k.

Memory Address and Capacity

A representation of a small 8 * 8 memory chip is shown in Figure 2(a). The location of a unit of data in a memory array is called its **address**. For example, in part (b), the address of a bit in the 2-dimensional array is specified by the row and column as shown. In part (c), the address of a byte is specified only by the row. So, as you can see, the address depends on how the memory is organized into units of data. Personal computers have random access memories organized in bytes. This means that the smallest group of bits that can be addressed is eight.



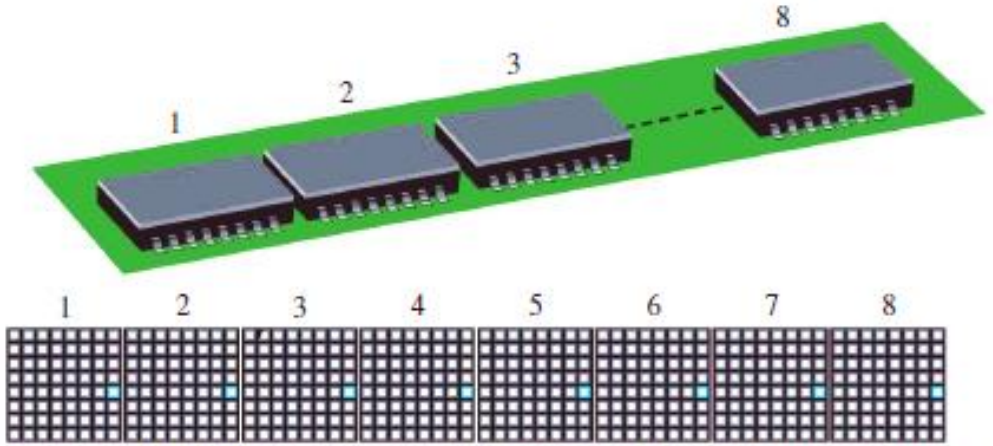
(a) Physical structure of 64-bit memory.

(b) The address of the blue bit is row 5, column 4.

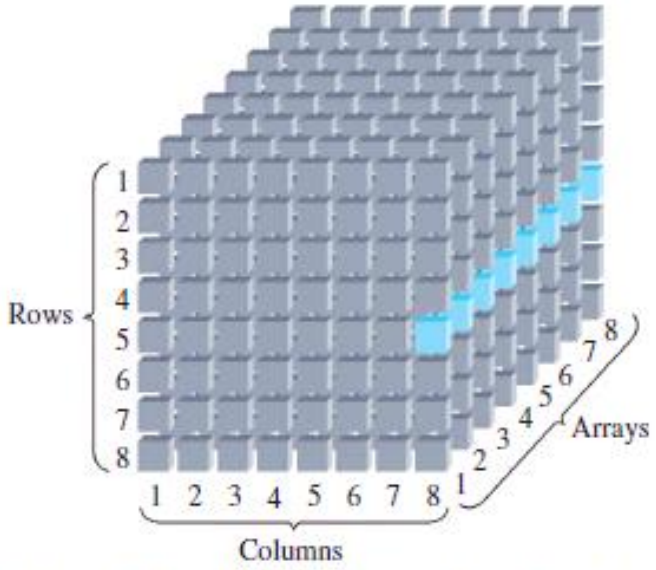
(c) The address of the blue byte is row 3.

FIGURE –2 Examples of memory address in a 2-dimensional memory array.

Figure 3(a) illustrates the expansion of the 8 * 8 (64-bit) array to a 64-byte memory. The address of a byte in the array is specified by the row and column, as shown. In this case, the smallest group of bits that can be accessed is eight. This can be viewed as a 3-dimensional array, as shown in part (b).



(a) The 8 × 8 bit array expanded to a 64 × 8 bit array. This array forms a memory module.



(b) The address of the blue byte is row 5, column 8.

FIGURE 3 Example of memory address in an expanded (multiple) array.

The **capacity** of a memory is the total number of data units that can be stored. For example, in the bit organized memory array in Figure 2(b), the capacity is 64 bits. In the byte-organized memory array in Figure 2(c), the capacity is 8 bytes, which is also 64 bits. In Figure 3, the capacity is 64 bytes. Computer memories typically have multiple gigabytes of internal memory. Computers usually transfer and store data as 64-bit words, in which case all eight bits of row five in each chip in Figure 3(a) would be accessed.

Memory Banks and Ranks

A **bank** is a section of memory within a single memory array (chip). A memory chip may have one or more banks. Memory banks can be used for storing frequently used information. Easier and faster access can be achieved by knowing the section of memory in which the data are stored. A **rank** is a group of chips that make up a memory module that stores data in units such as words or bytes. These terms are illustrated in Figure 4.

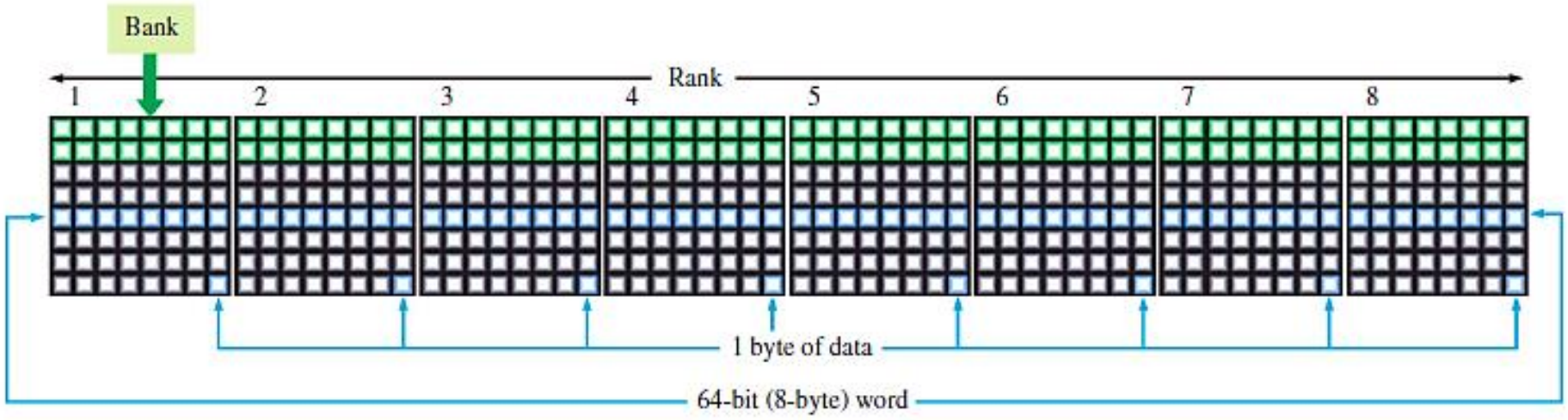


Figure 4 Simple illustration of memory bank and memory rank.

Basic Memory Operations

Addressing is the process of accessing a specified location in memory. Since a memory stores binary data, data must be put into the memory and data must be copied from the memory when needed. The **write** operation puts data into a specified address in the memory, and the **read** operation copies data out of a specified address in the memory. The addressing operation, which is part of both the write and the read operations, selects the specified memory address.

Data units go into the memory during a write operation and come out of the memory during a read operation on a set of lines called the *data bus.* As indicated in Figure 5,the data bus is bidirectional, which means that data can go in either direction (into the memory or out of the memory). In this case of byte-organized memories, the data bus has at least eight lines so that all eight bits in a selected address are transferred in parallel.

For a write or a read operation, an address is selected by placing a binary code representing the desired address on a set of lines called the *address bus*. The address code is decoded internally, and the appropriate address is selected. In the case of the multiple-array memory in Figure 5(b) there are two decoders, one for the rows and one for the columns. The number of lines in the address bus depends on the capacity of the memory. For example, a 15-bit address code can select 32,768 locations (2^{15}) in the memory, a 16-bit address code can select 65,536 locations (2^{16}) in the memory, and so on. In personal computers a 32-bit address bus can select 4,294,967,296 locations (2^{32}), expressed as 4G.

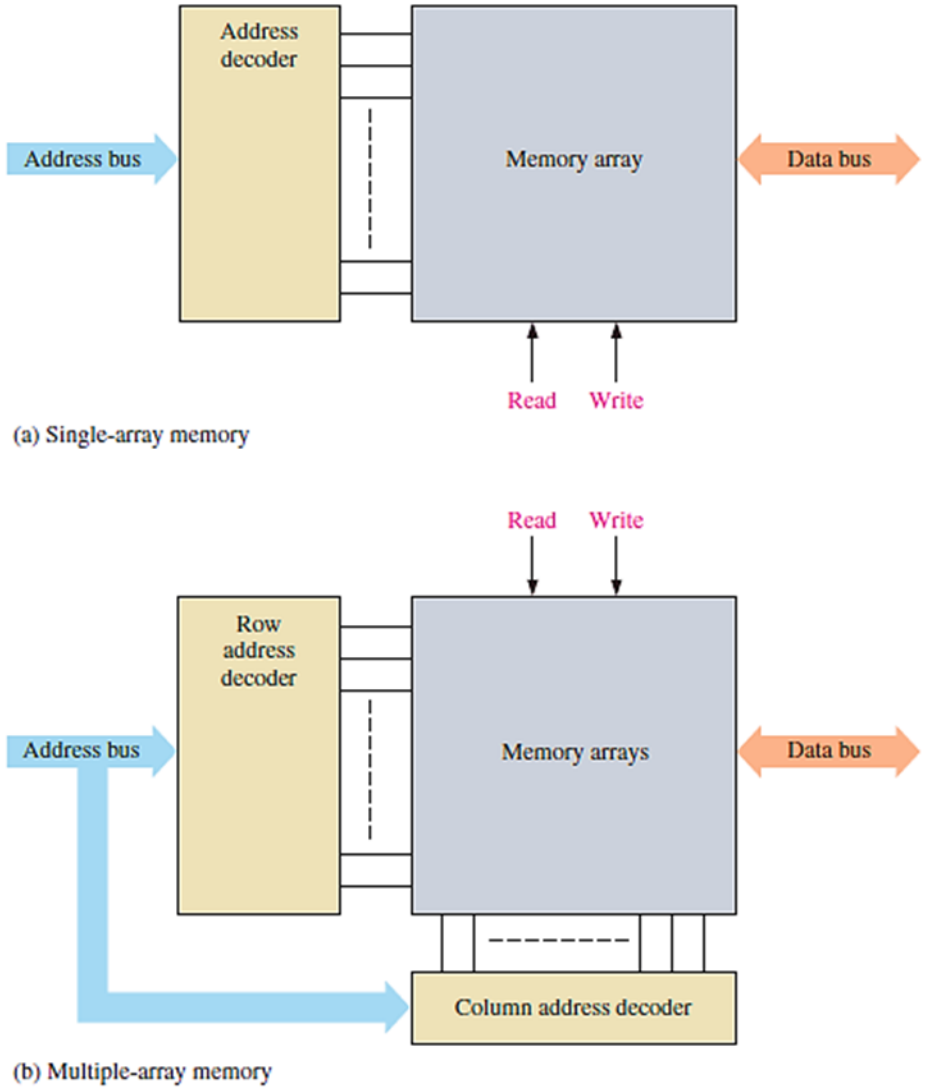


FIGURE 5 Block diagram of a single-array memory and a multiple-array memory showing address bus, address decoder(s), bidirectional data bus, and read/write inputs.

The Write Operation

A simplified write operation is illustrated in Figure 6. To store a byte of data in the memory, a code held in the address register is placed on the address bus. Once the address code is on the bus, the address decoder decodes the address and selects the specified location in the memory. The memory then gets a write command, and the data byte held in the data register is placed on the data bus and stored in the selected memory address, thus completing the write operation. When a new data byte is written into a memory address, the current data byte stored at that address is overwritten (replaced with a new data byte).

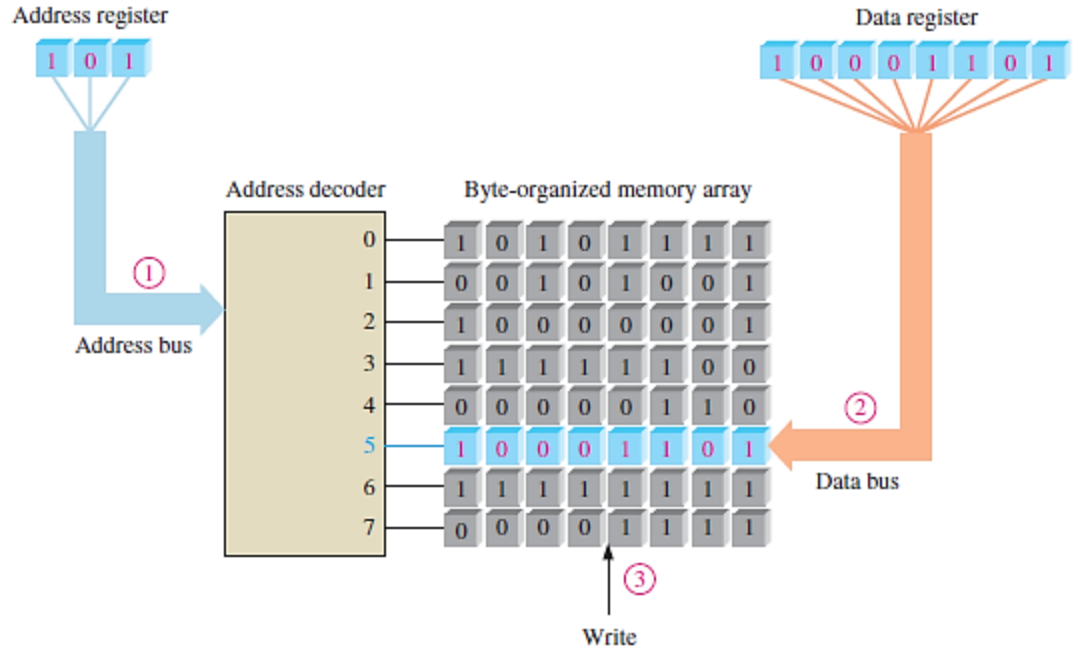


FIGURE 6 Illustration of the write operation.

① Address code 101 is placed on the address bus and address 5 is selected.

② Data byte is placed on the data bus.

③ Write command causes the data byte to be stored in address 5, replacing previous data.

The Read Operation

A simplified read operation is illustrated in Figure 7. Again, a code held in the address register is placed on the address bus. Once the address code is on the bus, the address decoder decodes the address and selects the specified location in the memory. The memory then gets a read command, and a "copy" of the data byte that is stored in the selected memory address is placed on the data bus and loaded into the data register, thus completing the read operation. When a data byte is read from a memory address, it also remains stored at that address. This is called *nondestructive read*.

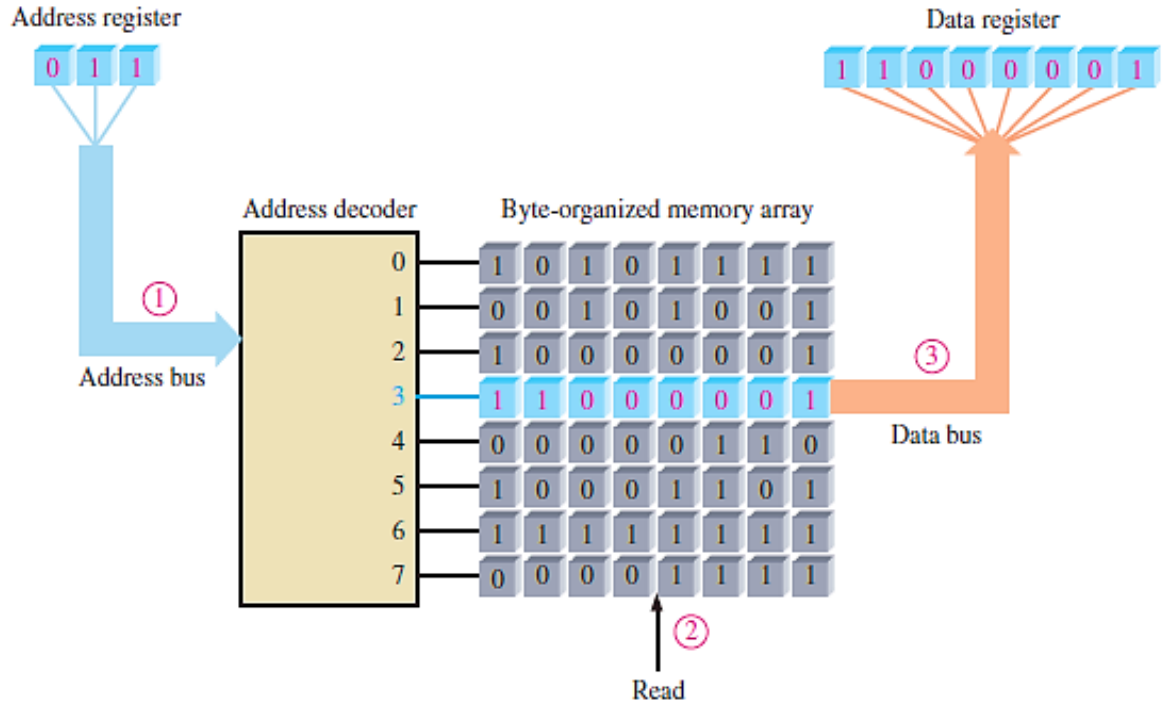


FIGURE 7 Illustration of the read operation.

① Address code 011 is placed on the address bus and address 3 is selected.
② Read command is applied.
③ The contents of address 3 is placed on the data bus and shifted into data register. The contents of address 3 is not erased by the read operation.

RAMs and ROMs

The two major categories of semiconductor memories are the RAM and the ROM. **RAM** (random-access memory) is a type of memory in which all addresses are accessible in an equal amount of time and can be selected in any order for a read or write operation. All RAMs have both *read* and *write* capability. Because RAMs lose stored data when the power is turned off, they are **volatile** memories.

ROM (read-only memory) is a type of memory in which data are stored permanently or semipermanently. Data can be read from a ROM, but there is no write operation as in the RAM. The ROM, like the RAM, is a random-access memory but the term *RAM* traditionally means a random-access *read/write* memory. Several types of RAMs and ROMs will be covered in this chapter. Because ROMs retain stored data even if power is turned off, they are **nonvolatile** memories.

Thank You