CHAPTER 11

Data Storage

CHAPTER OUTLINE

CHAPTER OBJECTIVES

- Define the basic memory characteristics
- Explain what a RAM is and how it works
- Explain the difference between static RAMs (SRAMs) and dynamic RAMs (DRAMs)
- Explain what a ROM is and how it works
- Describe the various types of PROMs
- Discuss the characteristics of a flash memory
- Describe the expansion of ROMs and RAMs to increase word length and word capacity
- Discuss special types of memories such as FIFO and LIFO
- Describe the basic organization of magnetic disks and magnetic tapes
- Describe the basic operation of magneto-optical disks and optical disks
- Describe the key elements in a memory hierarchy
- Describe several characteristics of cloud storage
- Describe basic methods for memory testing
- Develop flowcharts for memory testing

KEY TERMS

Key terms are in order of appearance in the chapter.

- Memory
- Byte
- Word
- Cell
- Address
- Capacity
- Write
- Read
- RAM
- ROM
- SRAM
- DRAM
- Bus
- PROM
- EPROM
- Flash memory
- FIFO
- LIFO
- Hard disk
- Blu-ray
- Memory hierarchy
- Cloud storage
- Server

VISIT THE WEBSITE

Study aids for this chapter are available at http://www.pearsonglobaleditions.com/floyd

INTRODUCTION

Chapter 8 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.

11–1 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, so we will use smaller ones for illustration in this chapter to simplify the concepts.

After completing this chapter, you should be able to

- Explain how a memory stores binary data
- Discuss the basic organization of a memory
- Describe the write operation
- Describe the read operation
- Describe the addressing operation
- Explain what RAMs and ROMs are

InfoNote

The general definition of *word* is a complete unit of information consisting of a unit of binary data. When applied to computer instructions, a word is more specifically defined as two bytes (16 bits). As an important part of assembly language used in computers, the DW (Define Word) directive means to define data in 16-bit units. This definition is independent of the particular microprocessor or the size of its data bus. Assembly language also allows definitions of bytes (8 bits) with the DB directive, double words (32 bits) with the DD directive, and quad-words (64 bits) with the QD directive.

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**. Memories are made up of arrays of cells, as illustrated in Figure 11–1 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.

The 64-cell array can be organized in several ways based on units of data. Figure 11–1(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

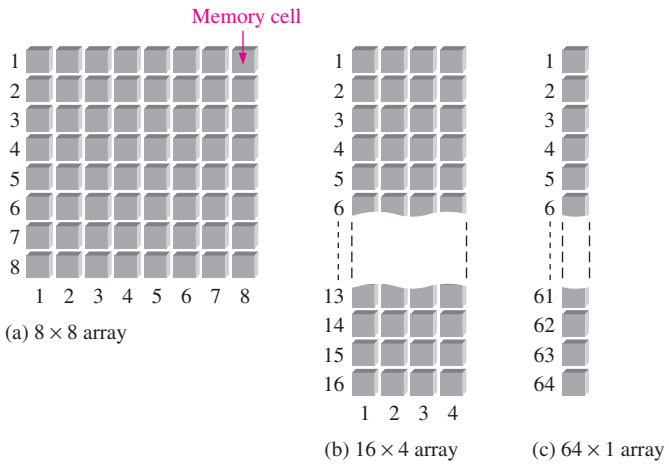
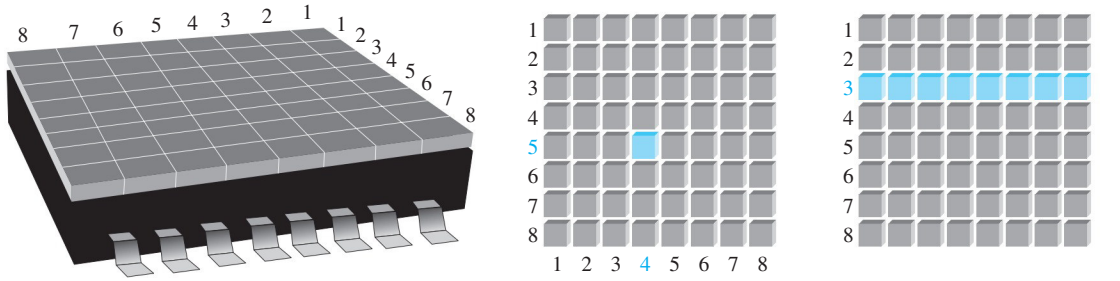


FIGURE 11–1 A 64-cell memory array organized in three different ways.

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 11–2(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.



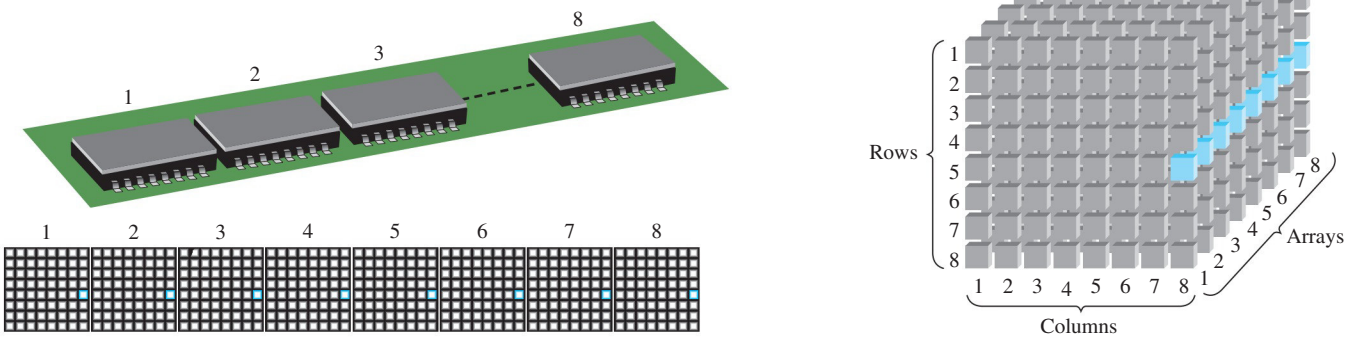
(a) Physical structure of 64-bit memory.

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

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

FIGURE 11–2 Examples of memory address in a 2-dimensional memory array.

Figure 11–3(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 11–3 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 11–2(b), the capacity is 64 bits. In the byte-organized memory array in Figure 11–2(c), the capacity is 8 bytes, which is also

64 bits. In Figure 11–3, 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 11–3(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 11–4.

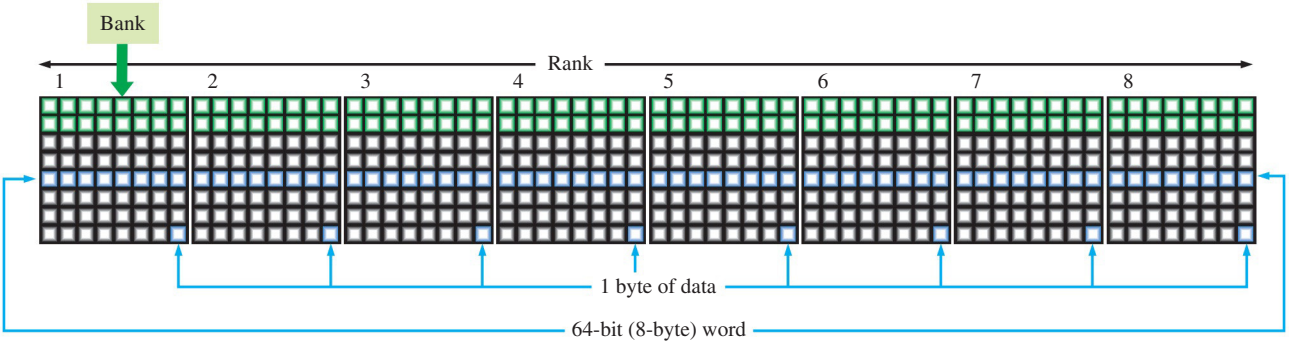


FIGURE 11–4 Simple illustration of memory bank and memory rank.

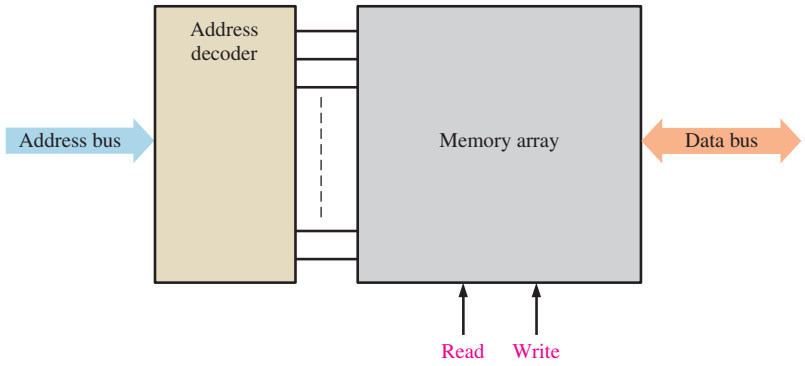
Basic Memory Operations

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.

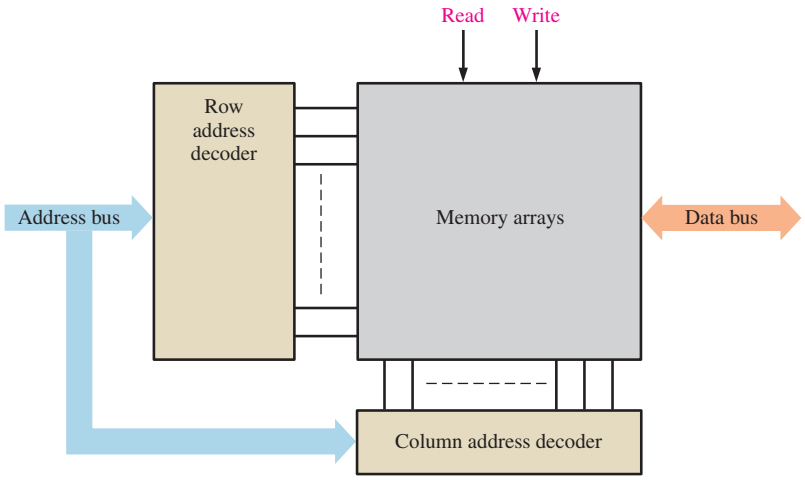
Data units go into the memory during a write operation and come out of the memory during a read operation on a set of lines called the *data bus.* As indicated in Figure 11–5, the data bus is bidirectional, which means that data can go in either direction (into the memory or out of the memory). In this 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 11–5(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.

The Write Operation

A simplified write operation is illustrated in Figure 11–6. 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).

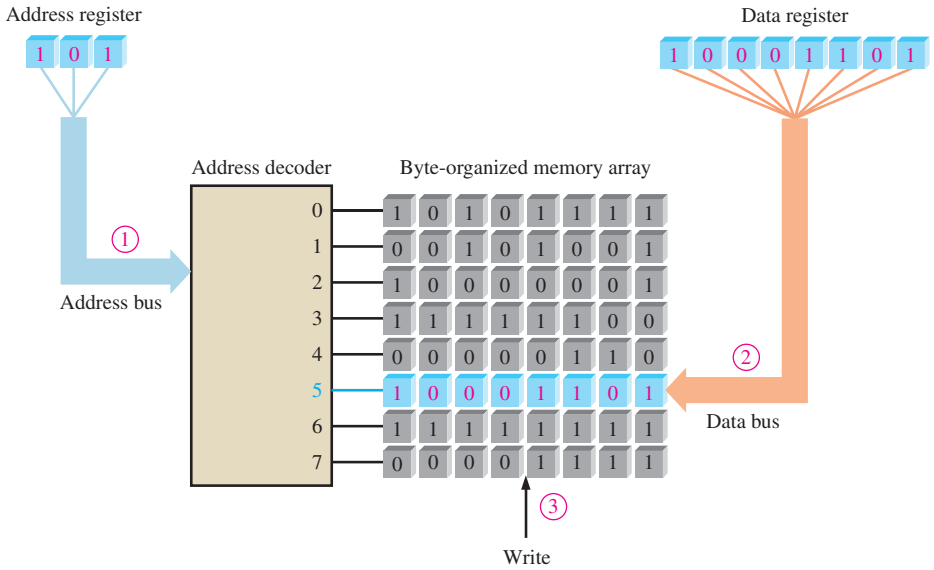


(a) Single-array memory



(b) Multiple-array memory

FIGURE 11–5 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.

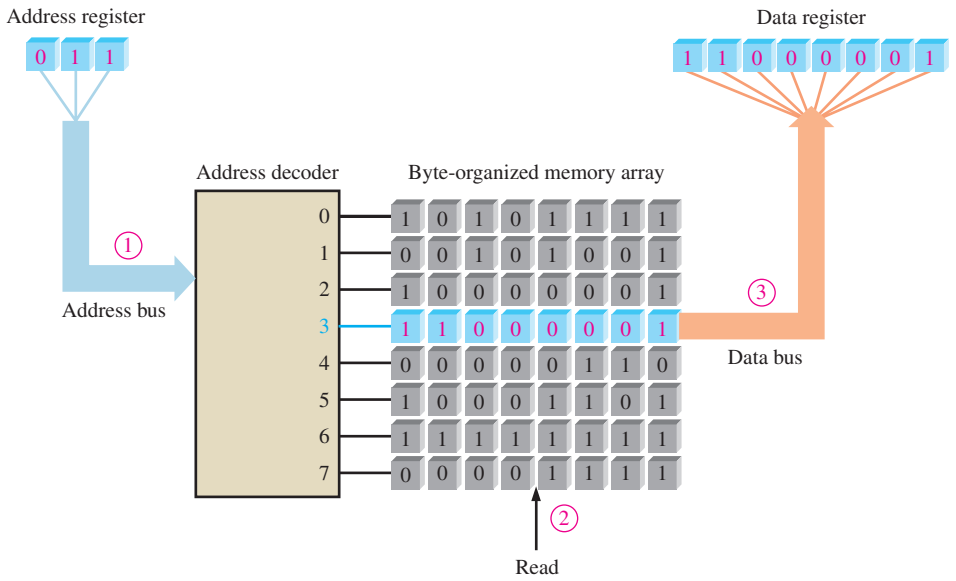


1. Address code 101 is placed on the address bus and address 5 is selected.
2. Data byte is placed on the data bus.
3. Write command causes the data byte to be stored in address 5, replacing previous data.

FIGURE 11–6 Illustration of the write operation.

The Read Operation

A simplified read operation is illustrated in Figure 11–7. 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.*



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.

FIGURE 11–7 Illustration of 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.

SECTION 11–1 CHECKUP

Answers are at the end of the chapter.

1. What is the smallest unit of data that can be stored in a memory?
2. What is the bit capacity of a memory that can store 256 bytes of data?

3. What is a write operation?
4. What is a read operation?
5. How is a given unit of data located in a memory?
6. Describe the difference between a RAM and a ROM.

11–2 The Random-Access Memory (RAM)

A RAM is a read/write memory in which data can be written into or read from any selected address in any sequence. When a data unit is written into a given address in the RAM, the data unit previously stored at that address is replaced by the new data unit. When a data unit is read from a given address in the RAM, the data unit remains stored and is not erased by the read operation. This nondestructive read operation can be viewed as copying the content of an address while leaving the content intact. A RAM is typically used for short-term data storage because it cannot retain stored data when power is turned off.

After completing this section, you should be able to

- Name the two categories of RAM
- Explain what a SRAM is
- Describe the SRAM storage cell
- Explain the difference between an asynchronous SRAM and a synchronous burst SRAM
- Explain the purpose of a cache memory
- Explain what a DRAM is
- Describe the DRAM storage cells
- Discuss the types of DRAM
- Compare the SRAM with the DRAM

The RAM Family

The two major categories of RAM are the *static RAM* (SRAM) and the *dynamic RAM* (DRAM). **SRAMs** generally use latches as storage elements and can therefore store data indefinitely *as long as dc power is applied.* **DRAMs** use capacitors as storage elements and cannot retain data very long without the capacitors being recharged by a process called **refreshing.** Both SRAMs and DRAMs will lose stored data when dc power is removed and, therefore, are classified as volatile memories.

Data can be read much faster from SRAMs than from DRAMs. However, DRAMs can store much more data than SRAMs for a given physical size and cost because the DRAM cell is much simpler and more cells can be crammed into a given chip area than in the SRAM.

The basic types of SRAM are the *asynchronous SRAM* and the *synchronous SRAM* with a burst feature. The basic types of DRAM are the *Fast Page Mode DRAM* (FPM DRAM), the *Extended Data Out DRAM* (EDO DRAM), the *Burst EDO DRAM* (BEDO DRAM), and the *synchronous DRAM* (SDRAM). These are shown in Figure 11–8.

Static RAMs (SRAMs)

Memory Cell

All SRAMs are characterized by latch memory cells. As long as dc power is applied to a **static memory** cell, it can retain a 1 or 0 state indefinitely. If power is removed, the stored data bit is lost.

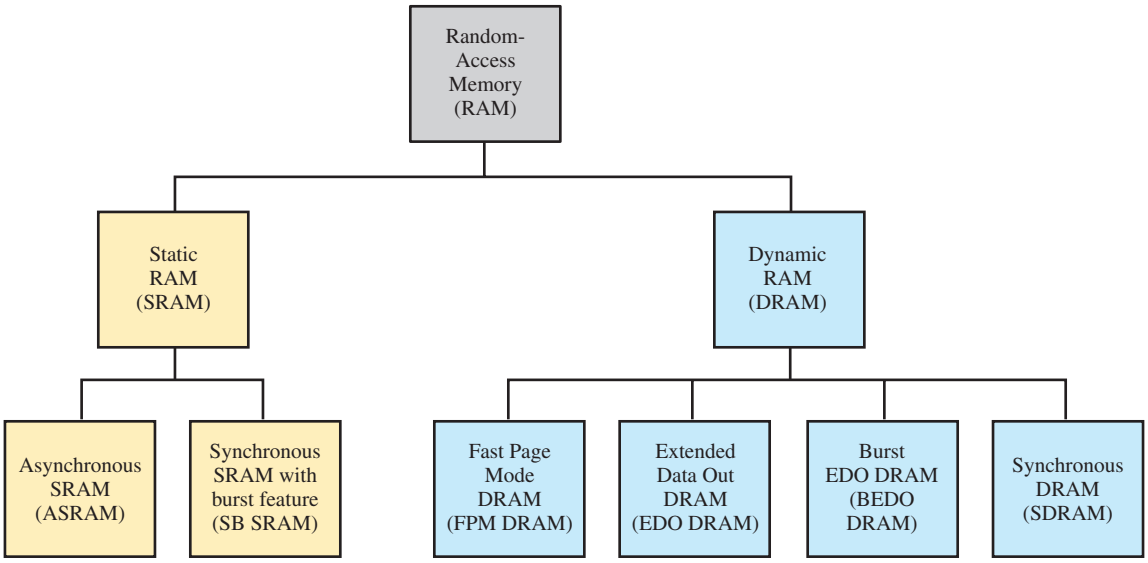


FIGURE 11–8 The RAM family.

Figure 11–9 shows a basic SRAM latch memory cell. The cell is selected by an active level on the Select line and a data bit (1 or 0) is written into the cell by placing it on the Data in line. A data bit is read by taking it off the Data out line.

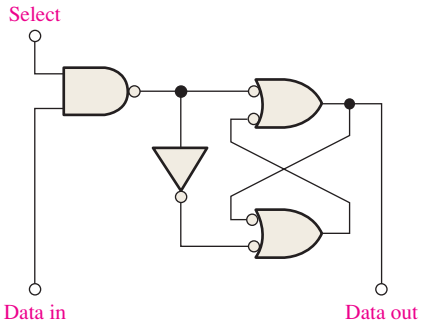


FIGURE 11–9 A typical SRAM latch memory cell.

Static Memory Cell Array

The memory cells in a SRAM are organized in rows and columns, as illustrated in Figure 11–10 for the case of an $n \times 4$ array. All the cells in a row share the same Row Select line. Each set of Data in and Data out lines go to each cell in a given column and are connected to a single data line that serves as both an input and output (Data I/O) through the data input and data output buffers.

To write a data unit, in this case a nibble (4 bits), into a given row of cells in the memory array, the Row Select line is taken to its active state and four data bits are placed on the Data I/O lines. The Write line is then taken to its active state, which causes each data bit to be stored in a selected cell in the associated column. To read a data unit, the Read line is taken to its active state, which causes the four data bits stored in the selected row to appear on the Data I/O lines.

Basic Asynchronous SRAM Organization

An asynchronous SRAM is one in which the operation is not synchronized with a system clock. To illustrate the general organization of a SRAM, a 32k $\times$ 8 bit memory is used. A logic symbol for this memory is shown in Figure 11–11.

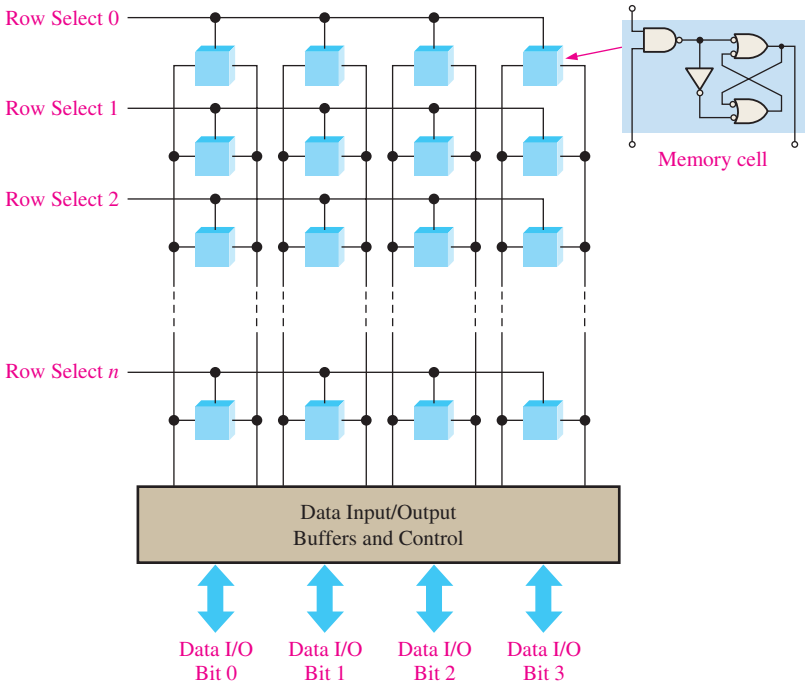


FIGURE 11–10 Basic SRAM array.

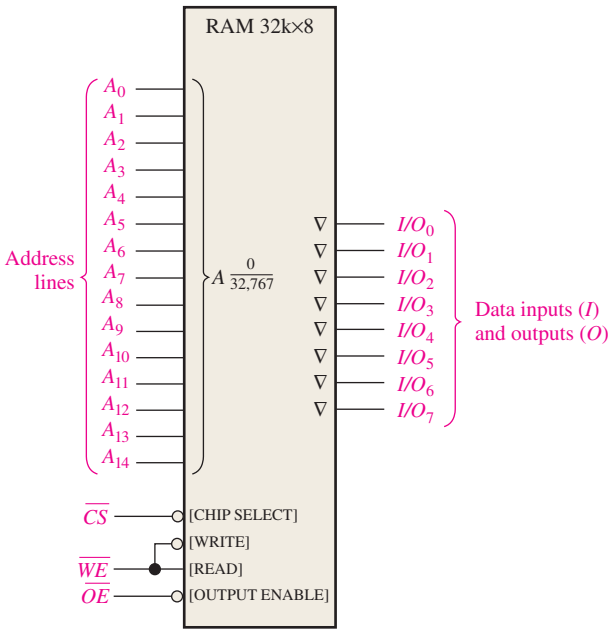


FIGURE 11–11 Logic diagram for an asynchronous 32k × 8 SRAM.

In the READ mode, the eight data bits that are stored in a selected address appear on the data output lines. In the WRITE mode, the eight data bits that are applied to the data input lines are stored at a selected address. The data input and data output lines (I/O_0 through I/O_7) share the same lines. During READ, they act as output lines (O_0 through O_7) and during WRITE they act as input lines (I_0 through I_7).

Tri-state Outputs and Buses

Tri-state buffers in a memory allow the data lines to act as either input or output lines and connect the memory to the data bus in a computer. These buffers have three output states:

HIGH (1), LOW (0), and HIGH-*Z* (open). Tri-state outputs are indicated on logic symbols by a small inverted triangle (∇), as shown in Figure 11–11, and are used for compatibility with bus structures such as those found in microprocessor-based systems.

Physically, a **bus** is one or more conductive paths that serve to interconnect two or more functional components of a system or several diverse systems. Electrically, a bus is a collection of specified voltage levels and/or current levels and signals that allow various devices to communicate and work properly together.

A microprocessor is connected to memories and input/output devices by certain bus structures. An address bus allows the microprocessor to address the memories, and a data bus provides for transfer of data between the microprocessor, the memories, and the input/output devices such as monitors, printers, keyboards, and modems. A control bus allows the microprocessor to control data transfers and timing for the various components.

Memory Array

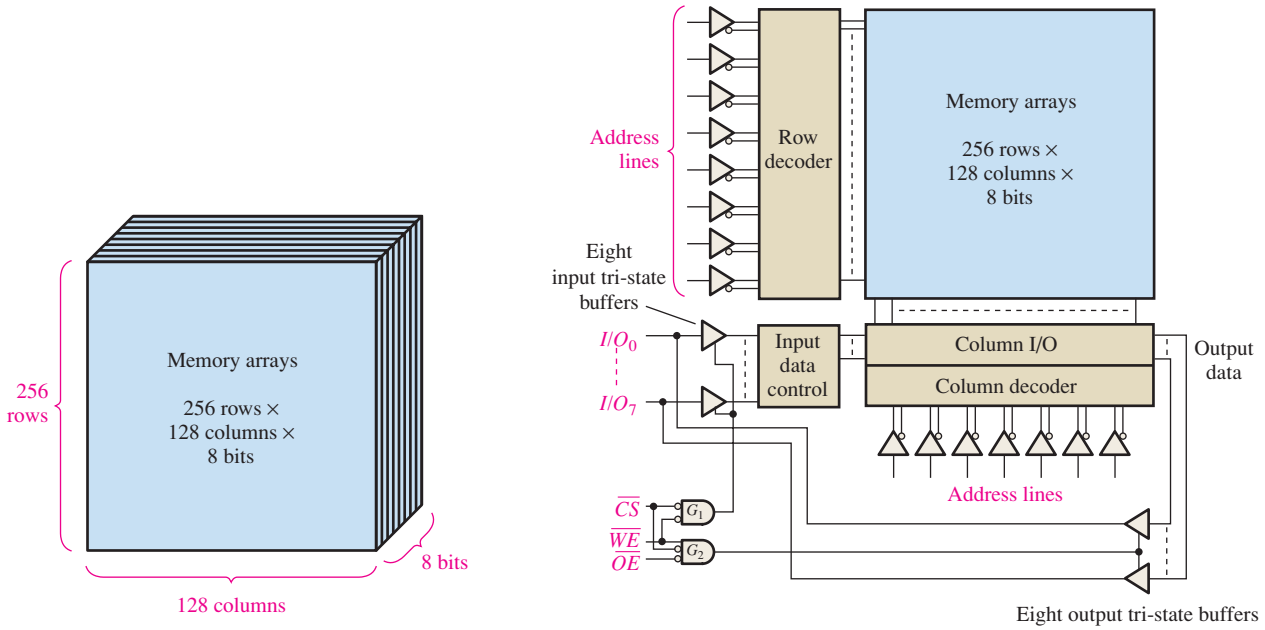
SRAM chips can be organized in single bits, nibbles (4 bits), bytes (8 bits), or multiple bytes (words with 16, 24, 32 bits, etc.).

Figure 11–12 shows the organization of a small 32k × 8 SRAM. The memory cell array is arranged in 256 rows and 128 columns, each with 8 bits, as shown in part (a). There are actually $2^{15} = 32{,}768$ addresses and each address contains 8 bits. The capacity of this example memory is 32,768 bytes (typically expressed as 32 kB). Although small by today's standards, this memory serves to introduce the basic concepts.

The SRAM in Figure 11–12(b) works as follows. First, the chip select, $\overline{CS}$, must be LOW for the memory to operate. (Other terms for chip select are *enable* or *chip enable.*) Eight of the fifteen address lines are decoded by the row decoder to select one of the 256 rows. Seven of the fifteen address lines are decoded by the column decoder to select one of the 128 8-bit columns.

Read

In the READ mode, the write enable input, $\overline{WE}$, is HIGH and the output enable, $\overline{OE}$, is LOW. The input tri-state buffers are disabled by gate G_1, and the column output tri-state



(a) Memory array configuration

(b) Memory block diagram

FIGURE 11–12 Basic organization of an asynchronous 32k × 8 SRAM.

buffers are enabled by gate G_2. Therefore, the eight data bits from the selected address are routed through the column I/O to the data lines (I/O_0 though I/O_7), which are acting as data output lines.

Write

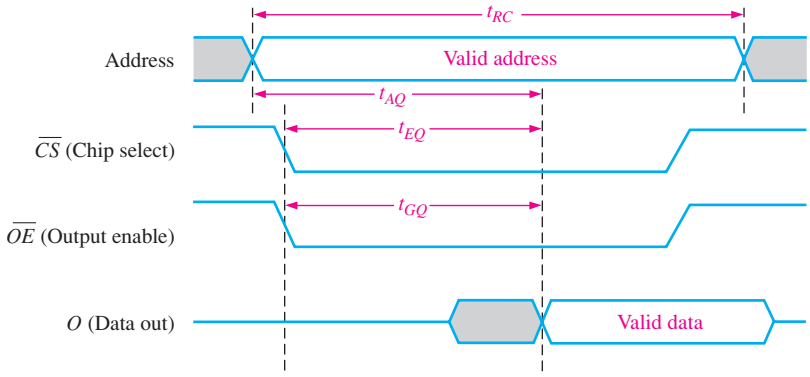
In the WRITE mode, $\overline{WE}$ is LOW and $\overline{OE}$ is HIGH. The input tri-state buffers are enabled by gate G_1, and the output tri-state buffers are disabled by gate G_2. Therefore, the eight input data bits on the data lines are routed through the input data control and the column I/O to the selected address and stored.

Read and Write Cycles

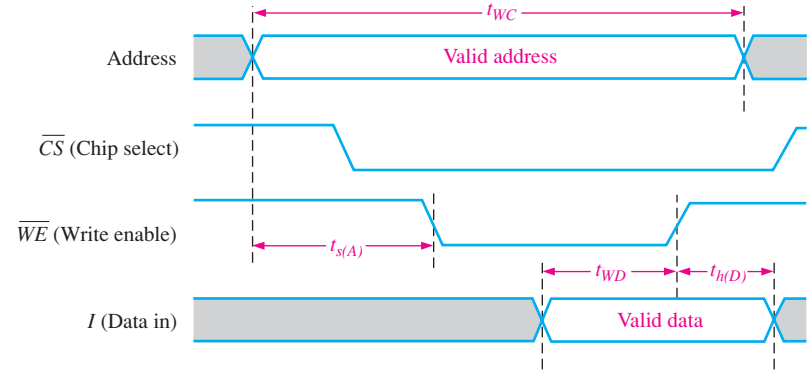
Figure 11–13 shows typical timing diagrams for a memory read cycle and a write cycle. For the read cycle shown in part (a), a valid address code is applied to the address lines for a specified time interval called the *read cycle time,* t_{RC}. Next, the chip select ($\overline{CS}$) and the output enable ($\overline{OE}$) inputs go LOW. One time interval after the $\overline{OE}$ input goes LOW, a valid data byte from the selected address appears on the data lines. This time interval is called the *output enable access time,* t_{GQ}. Two other access times for the read cycle are the *address access time,* t_{AQ}, measured from the beginning of a valid address to the appearance of valid data on the data lines and the *chip enable access time,* t_{EQ}, measured from the HIGH-to-LOW transition of $\overline{CS}$ to the appearance of valid data on the data lines.

During each read cycle, one unit of data, a byte in this case, is read from the memory.

For the write cycle shown in Figure 11–13(b), a valid address code is applied to the address lines for a specified time interval called the *write cycle time,* t_{WC}. Next, the chip



(a) Read cycle ($\overline{WE}$ HIGH)



(b) Write cycle ($\overline{WE}$ LOW)

FIGURE 11–13 Timing diagrams for typical read and write cycles for the SRAM in Figure 11–12.

select ($\overline{CS}$) and the write enable ($\overline{WE}$) inputs go LOW. The required time interval from the beginning of a valid address until the $\overline{WE}$ input goes LOW is called the *address setup time,* $t_{s(A)}$. The time that the $\overline{WE}$ input must be LOW is the write pulse width. The time that the input $\overline{WE}$ must remain LOW after valid data are applied to the data inputs is designated t_{WD}; the time that the valid input data must remain on the data lines after the $\overline{WE}$ input goes HIGH is the *data hold time,* $t_{h(D)}$.

During each write cycle, one unit of data is written into the memory.

Synchronous SRAM with Burst Feature

Unlike the asynchronous SRAM, a synchronous SRAM is synchronized with the system clock. For example, in a computer system, the synchronous SRAM operates with the same clock signal that operates the microprocessor so that the microprocessor and memory are synchronized for faster operation.

The fundamental concept of the synchronous feature of a SRAM can be shown with Figure 11–14, which is a simplified block diagram of a 32k × 8 memory for purposes of illustration. The synchronous SRAM is similar to the asynchronous SRAM in terms of the memory array, address decoder, and read/write and enable inputs. The basic difference is that the synchronous SRAM uses clocked registers to synchronize all inputs with the system clock. The address, the read/write input, the chip enable, and the input data are all latched into their respective registers on an active clock pulse edge. Once this information is latched, the memory operation is in sync with the clock.

For the purpose of simplification, a notation for multiple parallel lines or bus lines is introduced in Figure 11–14, as an alternative to drawing each line separately. A set of

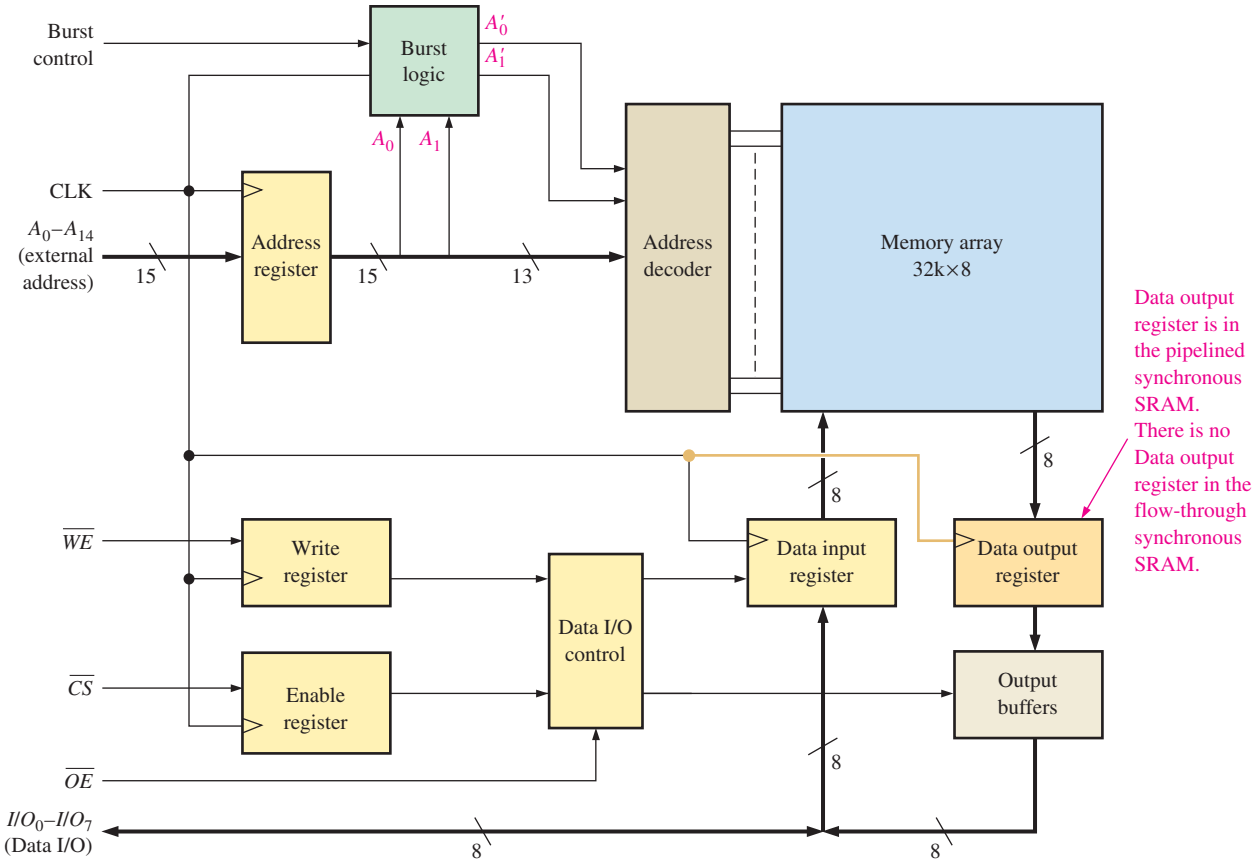
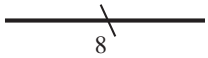


FIGURE 11–14 A basic block diagram of a synchronous SRAM with burst feature.

parallel lines can be indicated by a single heavy line with a slash and the number of separate lines in the set. For example, the following notation represents a set of 8 parallel lines:



The address bits A_0 through A_{14} are latched into the Address register on the positive edge of a clock pulse. On the same clock pulse, the state of the write enable ($\overline{WE}$) line and chip select ($\overline{CS}$) are latched into the Write register and the Enable register respectively. These are one-bit registers or simply flip-flops. Also, on the same clock pulse the input data are latched into the Data input register for a Write operation, and data in a selected memory address are latched into the Data output register for a Read operation, as determined by the Data I/O control based on inputs from the Write register, Enable register, and the Output enable ($\overline{OE}$).

Two basic types of synchronous SRAM are the *flow-through* and the *pipelined.* The flow-through synchronous SRAM does not have a Data output register, so the output data flow asynchronously to the data I/O lines through the output buffers. The **pipelined** synchronous SRAM has a Data output register, as shown in Figure 11–14, so the output data are synchronously placed on the data I/O lines.

The Burst Feature

As shown in Figure 11–14, synchronous SRAMs normally have an address burst feature, which allows the memory to read or write up to four sequential locations using a single address. When an external address is latched in the address register, the two lowest-order address bits, A_0 and A_1, are applied to the burst logic. This produces a sequence of four internal addresses by adding 00, 01, 10, and 11 to the two lowest-order address bits on successive clock pulses. The sequence always begins with the base address, which is the external address held in the address register.

The address burst logic in a typical synchronous SRAM consists of a binary counter and exclusive-OR gates, as shown in Figure 11–15. For 2-bit burst logic, the internal burst address sequence is formed by the base address bits A_2–A_{14} plus the two burst address bits A_1' and A_0'.

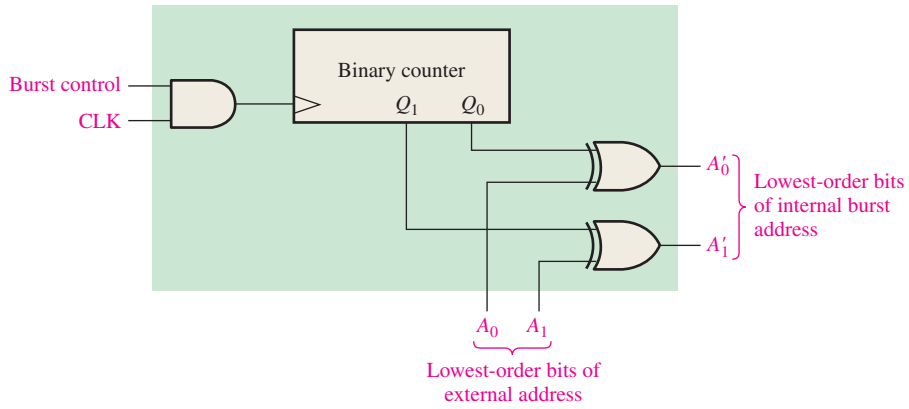


FIGURE 11–15 Address burst logic.

To begin the burst sequence, the counter is in its 00 state and the two lowest-order address bits are applied to the inputs of the XOR gates. Assuming that A_0 and A_1 are both 0, the internal address sequence in terms of its two lowest-order bits is 00, 01, 10, and 11.

Cache Memory

One of the major applications of SRAMs is in cache memories in computers. **Cache memory** is a relatively small, high-speed memory that stores the most recently used instructions or data from the larger but slower main memory. Cache memory can also use dynamic

RAM (DRAM), which is discussed next. Typically, SRAM is several times faster than DRAM. Overall, a cache memory gets stored information to the microprocessor much faster than if only high-capacity DRAM is used. Cache memory is basically a cost-effective method of improving system performance without having to resort to the expense of making all of the memory faster.

The concept of cache memory is based on the idea that computer programs tend to get instructions or data from one area of main memory before moving to another area. Basically, the cache controller "guesses" which area of the slow dynamic memory the CPU (central-processing unit) will need next and moves it to the cache memory so that it is ready when needed. If the cache controller guesses right, the data are immediately available to the microprocessor. If the cache controller guesses wrong, the CPU must go to the main memory and wait much longer for the correct instructions or data. Fortunately, the cache controller is right most of the time.

Cache Analogy

There are many analogies that can be used to describe a cache memory, but comparing it to a home refrigerator is perhaps the most effective. A home refrigerator can be thought of as a "cache" for certain food items while the supermarket is the main memory where all foods are kept. Each time you want something to eat or drink, you can go to the refrigerator (cache) first to see if the item you want is there. If it is, you save a lot of time. If it is not there, then you have to spend extra time to get it from the supermarket (main memory).

L1 and L2 Caches

A first-level cache (L1 cache) is usually integrated into the processor chip and has a very limited storage capacity. L1 cache is also known as *primary cache*. A second-level cache (L2 cache) may also be integrated into the processor or as a separate memory chip or set of chips external to the processor; it usually has a larger storage capacity than an L1 cache. L2 cache is also known as *secondary cache*. Some systems may have higher-level caches (L3, L4, etc.), but L1 and L2 are the most common. Also, some systems use a disk cache to enhance the performance of the hard disk because DRAM, although much slower than SRAM, is much faster than the hard disk drive. Figure 11–16 illustrates L1 and L2 cache memories in a computer system.

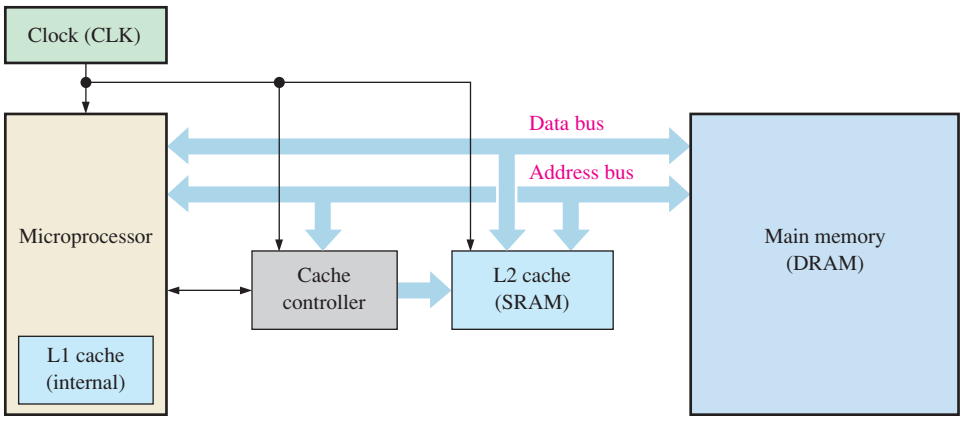


FIGURE 11–16 Block diagram showing L1 and L2 cache memories in a computer system.

Dynamic RAM (DRAM) Memory Cells

Dynamic memory cells store a data bit in a small capacitor rather than in a latch. The advantage of this type of cell is that it is very simple, thus allowing very large memory arrays to be constructed on a chip at a lower cost per bit. The disadvantage is that the

storage capacitor cannot hold its charge over an extended period of time and will lose the stored data bit unless its charge is refreshed periodically. To refresh requires additional memory circuitry and complicates the operation of the DRAM. Figure 11–17 shows a typical DRAM cell consisting of a single MOS transistor (MOSFET) and a capacitor.

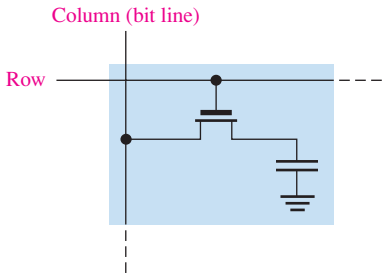


FIGURE 11–17 A MOS DRAM cell.

In this type of cell, the transistor acts as a switch. The basic simplified operation is illustrated in Figure 11–18 and is as follows. A LOW on the $R/\overline{W}$ line (WRITE mode) enables the tri-state input buffer and disables the output buffer. For a 1 to be written into the cell, the D_{IN} line must be HIGH, and the transistor must be turned on by a HIGH on the row line. The transistor acts as a closed switch connecting the capacitor to the bit line. This connection allows the capacitor to charge to a positive voltage, as shown in Figure 11–18(a). When a 0 is to be stored, a LOW is applied to the D_{IN} line. If the capacitor is storing a 0, it remains uncharged, or if it is storing a 1, it discharges as indicated in Figure 11–18(b). When the row line is taken back LOW, the transistor turns off and disconnects the capacitor from the bit line, thus "trapping" the charge (1 or 0) on the capacitor.

To read from the cell, the $R/\overline{W}$ (Read/$\overline{\text{Write}}$) line is HIGH, enabling the output buffer and disabling the input buffer. When the row line is taken HIGH, the transistor turns on and connects the capacitor to the bit line and thus to the output buffer (sense amplifier), so the data bit appears on the data-output line (D_{OUT}). This process is illustrated in Figure 11–18(c).

For refreshing the memory cell, the $R/\overline{W}$ line is HIGH, the row line is HIGH, and the refresh line is HIGH. The transistor turns on, connecting the capacitor to the bit line. The output buffer is enabled, and the stored data bit is applied to the input of the refresh buffer, which is enabled by the HIGH on the refresh input. This produces a voltage on the bit line corresponding to the stored bit, thus replenishing the capacitor. This is illustrated in Figure 11–18(d).

DRAM Organization

The major application of DRAMs is in the main memory of computers. The difference between DRAMs and SRAMs is the type of memory cell. As you have seen, the DRAM memory cell consists of one transistor and a capacitor and is much simpler than the SRAM cell. This allows much greater densities in DRAMs and results in greater bit capacities for a given chip area, although much slower access time.

Again, because charge stored in a capacitor will leak off, the DRAM cell requires a frequent refresh operation to preserve the stored data bit. This requirement results in more complex circuitry than in a SRAM. Several features common to most DRAMs are now discussed, using a generic 1M $\times$ 1 bit DRAM as an example.

Address Multiplexing

DRAMs use a technique called *address multiplexing* to reduce the number of address lines. Figure 11–19 shows the block diagram of a 1,048,576-bit (1 Mb) DRAM with a 1M $\times$ 1

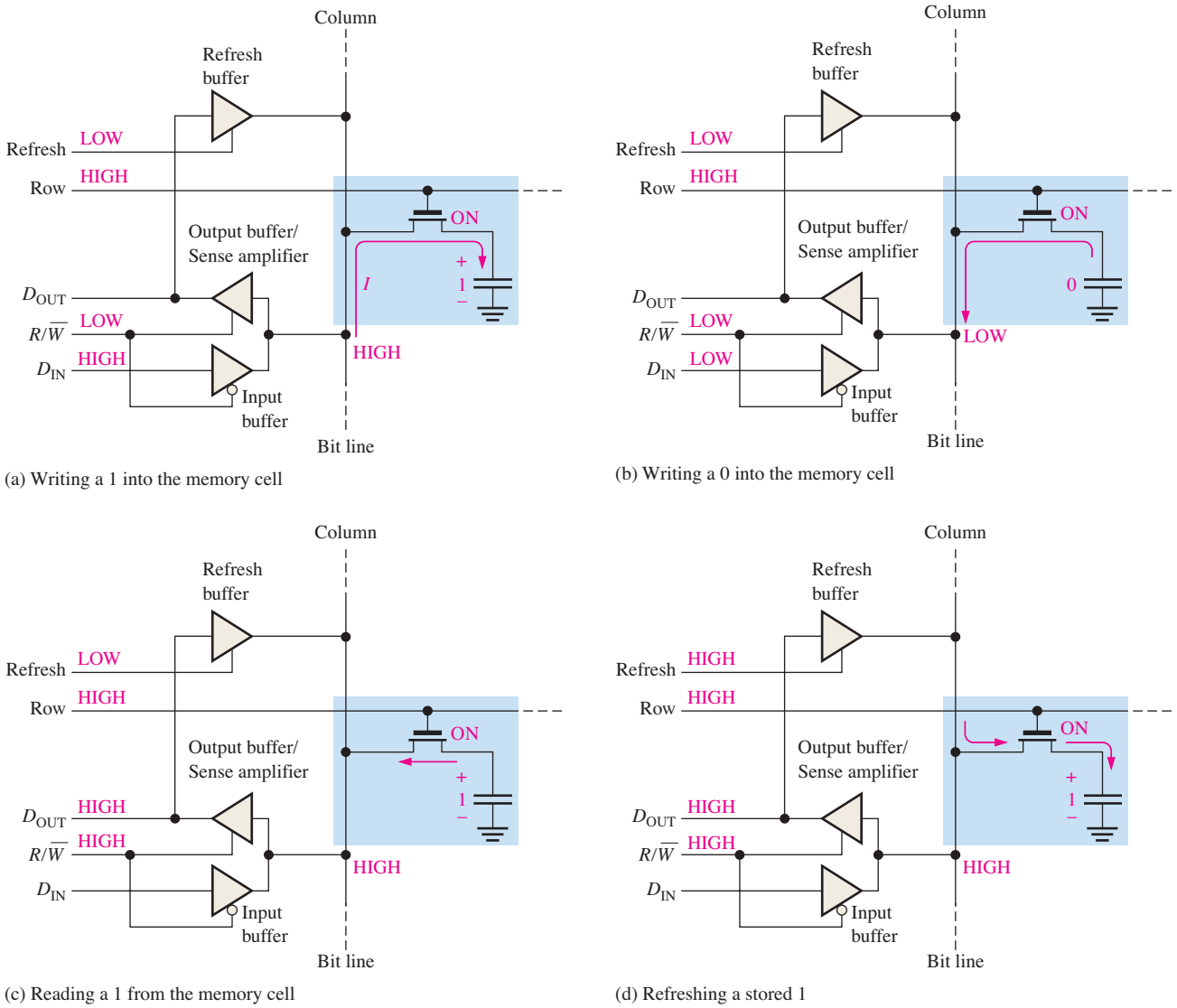


FIGURE 11–18 Basic operation of a DRAM cell.

organization. We will focus on the blue blocks to illustrate address multiplexing. The green blocks represent the refresh logic.

The ten address lines are time multiplexed at the beginning of a memory cycle by the row address select ($\overline{RAS}$) and the column address select ($\overline{CAS}$) into two separate 10-bit address fields. First, the 10-bit row address is latched into the row address register. Next, the 10-bit column address is latched into the column address register. The row address and the column address are decoded to select one of the 1,048,576 addresses ($2^{20} = 1{,}048{,}576$) in the memory array. The basic timing for the address multiplexing operation is shown in Figure 11–20.

Read and Write Cycles

At the beginning of each read or write memory cycle, $\overline{RAS}$ and $\overline{CAS}$ go active (LOW) to multiplex the row and column addresses into the registers, and decoders. For a read cycle, the $R/\overline{W}$ input is HIGH. For a write cycle, the $R/\overline{W}$ input is LOW. This is illustrated in Figure 11–21.

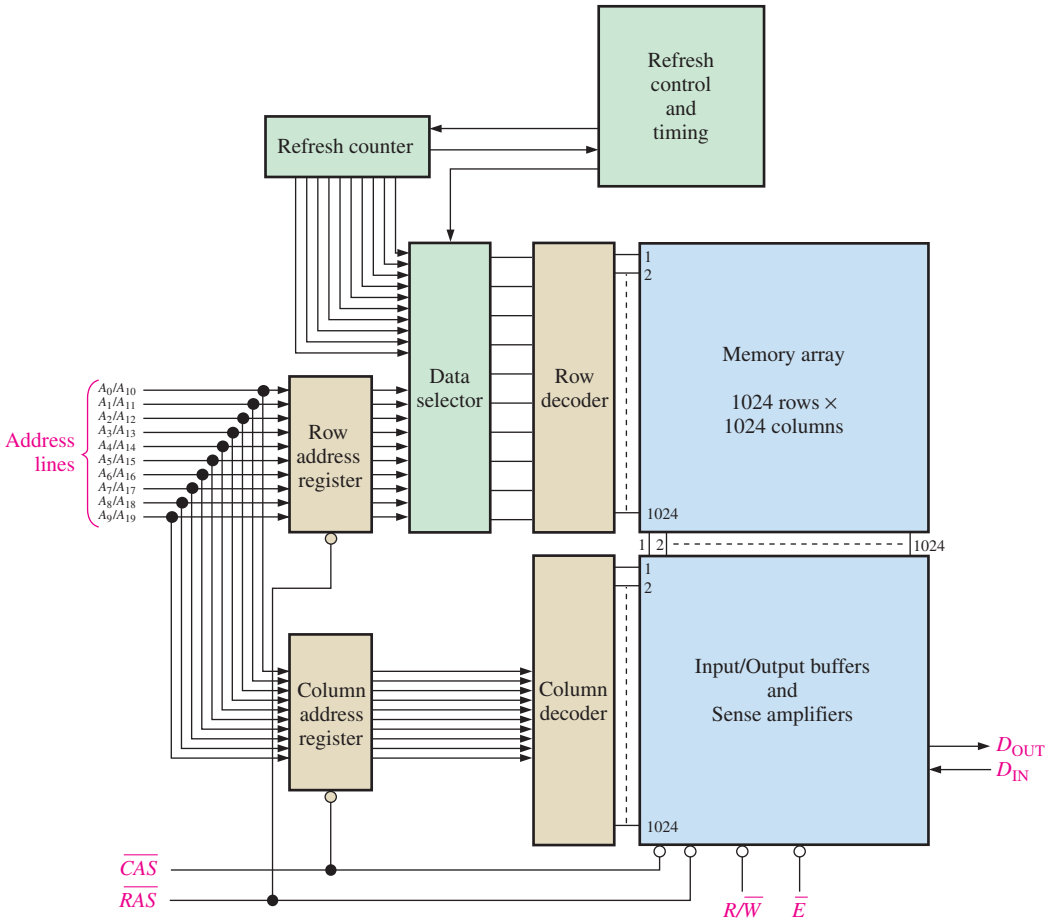


FIGURE 11–19 Simplified block diagram of a 1M × 1 DRAM.

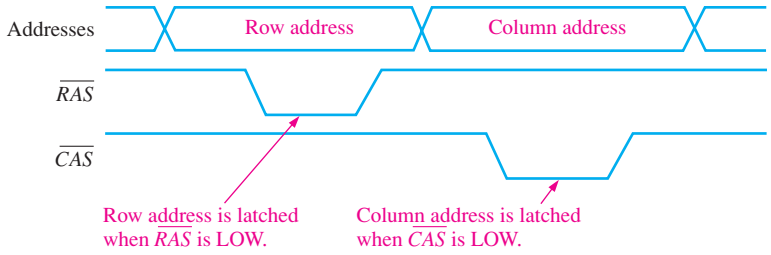


FIGURE 11–20 Basic timing for address multiplexing.

Fast Page Mode

In the normal read or write cycle described previously, the row address for a particular memory location is first loaded by an active-LOW $\overline{RAS}$ and then the column address for that location is loaded by an active-LOW $\overline{CAS}$. The next location is selected by another $\overline{RAS}$ followed by a $\overline{CAS}$, and so on.

A "page" is a section of memory available at a single row address and consists of all the columns in a row. Fast page mode allows fast successive read or write operations at each column address in a selected row. A row address is first loaded by $\overline{RAS}$ going LOW and remaining LOW while $\overline{CAS}$ is toggled between HIGH and LOW. A single row address is selected and remains selected while $\overline{RAS}$ is active. Each successive $\overline{CAS}$ selects another column in the selected row. So, after a fast page mode cycle, all of the addresses in the

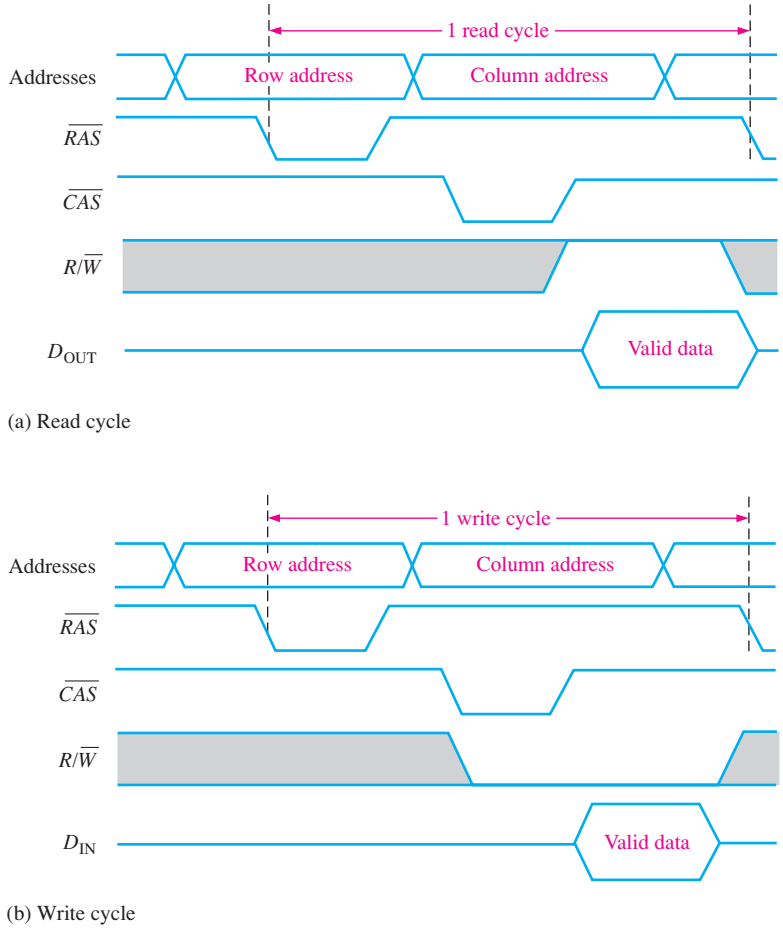


FIGURE 11–21 Timing diagrams for normal read and write cycles.

selected row have been read from or written into, depending on $R/\overline{W}$. For example, a fast page mode cycle for the DRAM in Figure 11–19 requires $\overline{CAS}$ to go active 1024 times for each row selected by $\overline{RAS}$.

Fast page mode operation for read is illustrated by the timing diagram in Figure 11–22. When $\overline{CAS}$ goes to its nonasserted state (HIGH), it disables the data outputs. Therefore, the transition of $\overline{CAS}$ to HIGH must occur only after valid data are latched by the external system.

Refresh Cycles

As you know, DRAMs are based on capacitor charge storage for each bit in the memory array. This charge degrades (leaks off) with time and temperature, so each bit must be periodically refreshed (recharged) to maintain the correct bit state. Typically, a DRAM must be refreshed every several milliseconds, although for some devices the refresh period can be much longer.

A read operation automatically refreshes all the addresses in the selected row. However, in typical applications, you cannot always predict how often there will be a read cycle, and so you cannot depend on a read cycle to occur frequently enough to prevent data loss. Therefore, special refresh cycles must be implemented in DRAM systems.

Burst refresh and *distributed refresh* are the two basic refresh modes for refresh operations. In burst refresh, all rows in the memory array are refreshed consecutively each refresh period. For a memory with a refresh period of 8 ms, a burst refresh of all rows occurs once every 8 ms. The normal read and write operations are suspended during a burst

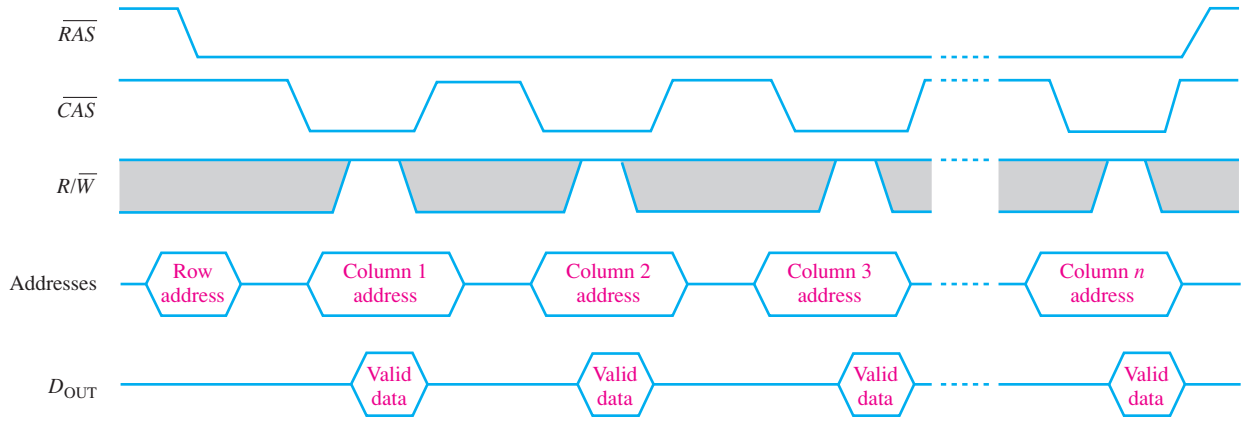


FIGURE 11–22 Fast page mode timing for a read operation.

refresh cycle. In distributed refresh, each row is refreshed at intervals interspersed between normal read or write cycles. For example, the memory in Figure 11–19 has 1024 rows. As an example, for an 8 ms refresh period, each row must be refreshed every 8 ms/1024 = 7.8 μs when distributed refresh is used.

The two types of refresh operations are $\overline{RAS}$ *only refresh* and $\overline{CAS}$ *before* $\overline{RAS}$ *refresh.* $\overline{RAS}$-only refresh consists of a $\overline{RAS}$ transition to the LOW (active) state, which latches the address of the row to be refreshed while $\overline{CAS}$ remains HIGH (inactive) throughout the cycle. An external counter is used to provide the row addresses for this type of operation.

The $\overline{CAS}$ before $\overline{RAS}$ refresh is initiated by $\overline{CAS}$ going LOW before $\overline{RAS}$ goes LOW. This sequence activates an internal refresh counter that generates the row address to be refreshed. This address is switched by the data selector into the row decoder.

Types of DRAMs

Now that you have learned the basic concept of a DRAM, let's briefly look at the major types. These are the *Fast Page Mode (FPM) DRAM,* the *Extended Data Out (EDO) DRAM,* the *Burst Extended Data Out (BEDO) DRAM,* and the *Synchronous (S) DRAM.*

FPM DRAM

Fast page mode operation was described earlier. Recall that a page in memory is all of the column addresses contained within one row address.

The idea of the **FPM DRAM** is based on the probability that the next several memory addresses to be accessed are in the same row (on the same page). Fortunately, this happens a large percentage of the time. FPM saves time over pure random accessing because in FPM the row address is specified only once for access to several successive column addresses whereas for pure random accessing, a row address is specified for each column address.

Recall that in a fast page mode read operation, the $\overline{CAS}$ signal has to wait until the valid data from a given address are accepted (latched) by the external system (CPU) before it can go to its nonasserted state. When $\overline{CAS}$ goes to its nonasserted state, the data outputs are disabled. This means that the next column address cannot occur until after the data from the current column address are transferred to the CPU. This limits the rate at which the columns within a page can be addressed.

EDO DRAM

The Extended Data Out DRAM, sometimes called *hyper page mode DRAM,* is similar to the FPM DRAM. The key difference is that the $\overline{CAS}$ signal in the **EDO DRAM** does not disable the output data when it goes to its nonasserted state because the valid data from the

current address can be held until $\overline{CAS}$ is asserted again. This means that the next column address can be accessed before the external system accepts the current valid data. The idea is to speed up the access time.

BEDO DRAM

The Burst Extended Data Out DRAM is an EDO DRAM with address burst capability. Recall from the discussion of the synchronous burst SRAM that the address burst feature allows up to four addresses to be internally generated from a single external address, which saves some access time. This same concept applies to the **BEDO DRAM.**

SDRAM

Faster DRAMs are needed to keep up with the ever-increasing speed of microprocessors. The Synchronous DRAM is one way to accomplish this. Like the synchronous SRAM discussed earlier, the operation of the **SDRAM** is synchronized with the system clock, which also runs the microprocessor in a computer system. The same basic ideas described in relation to the synchronous burst SRAM, also apply to the SDRAM.

This synchronized operation makes the SDRAM totally different from the other asynchronous DRAM types. With asynchronous memories, the microprocessor must wait for the DRAM to complete its internal operations. However, with synchronous operation, the DRAM latches addresses, data, and control information from the processor under control of the system clock. This allows the processor to handle other tasks while the memory read or write operations are in progress, rather than having to wait for the memory to do its thing as is the case in asynchronous systems.

DDR SDRAM

DDR stands for double data rate. A DDR SDRAM is clocked on both edges of a clock pulse, whereas a SDRAM is clocked on only one edge. Because of the double clocking, a DDR SDRAM is theoretically twice as fast as an SDRAM. Sometimes the SDRAM is referred to as an SDR SDRAM (single data rate SDRAM) for contrast with the DDR SDRAM.

SECTION 11–2 CHECKUP

1. List two types of SRAM.
2. What is a cache?
3. Explain how SRAMs and DRAMs differ.
4. Describe the refresh operation in a DRAM.
5. List four types of DRAM.

11–3 The Read-Only Memory (ROM)

A ROM contains permanently or semipermanently stored data, which can be read from the memory but either cannot be changed at all or cannot be changed without specialized equipment. A ROM stores data that are used repeatedly in system applications, such as tables, conversions, or programmed instructions for system initialization and operation. ROMs retain stored data when the power is off and are therefore nonvolatile memories.

After completing this section, you should be able to

- List the types of ROMs
- Describe a basic mask ROM storage cell
- Explain how data are read from a ROM
- Discuss internal organization of a typical ROM

The ROM Family

Figure 11–23 shows how semiconductor ROMs are categorized. The mask ROM is the type in which the data are permanently stored in the memory during the manufacturing process. The **PROM,** or programmable ROM, is the type in which the data are electrically stored by the user with the aid of specialized equipment. Both the mask ROM and the PROM can be of either MOS or bipolar technology. The **EPROM,** or erasable PROM, is strictly a MOS device. The **UV EPROM** is electrically programmable by the user, but the stored data must be erased by exposure to ultraviolet light over a period of several minutes. The electrically erasable PROM (**EEPROM** or E^2PROM) can be erased in a few milliseconds. The UV EPROM has been largely displaced by the EEPROM.

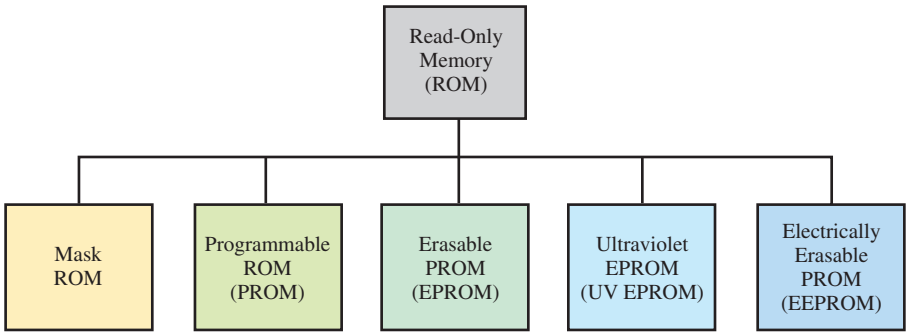


FIGURE 11–23 The ROM family.

The Mask ROM

The mask ROM is usually referred to simply as a ROM. It is permanently programmed during the manufacturing process to provide widely used standard functions, such as popular conversions, or to provide user-specified functions. Once the memory is programmed, it cannot be changed. Most IC ROMs utilize the presence or absence of a transistor connection at a row/column junction to represent a 1 or a 0.

Figure 11–24 shows MOS ROM cells. The presence of a connection from a row line to the gate of a transistor represents a 1 at that location because when the row line is taken HIGH, all transistors with a gate connection to that row line turn on and connect the HIGH (1) to the associated column lines. At row/column junctions where there are no gate connections, the column lines remain LOW (0) when the row is addressed.

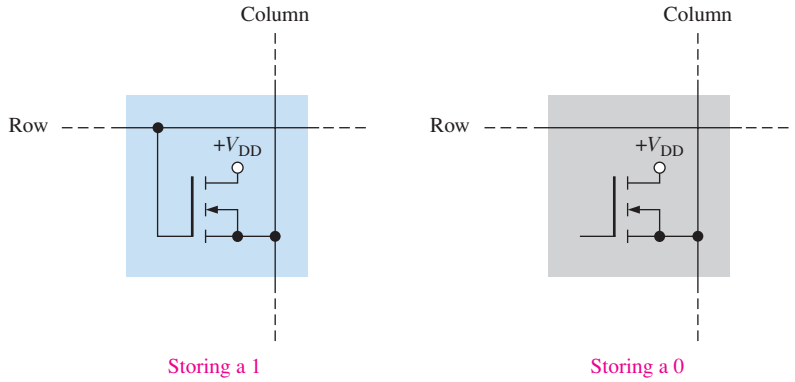


FIGURE 11–24 ROM cells.

To illustrate the ROM concept, Figure 11–25 shows a small, simplified ROM array. The blue squares represent stored 1s, and the gray squares represent stored 0s. The basic read operation is as follows. When a binary address code is applied to the address input lines, the

corresponding row line goes HIGH. This HIGH is connected to the column lines through the transistors at each junction (cell) where a 1 is stored. At each cell where a 0 is stored, the column line stays LOW because of the terminating resistor. The column lines form the data output. The eight data bits stored in the selected row appear on the output lines.

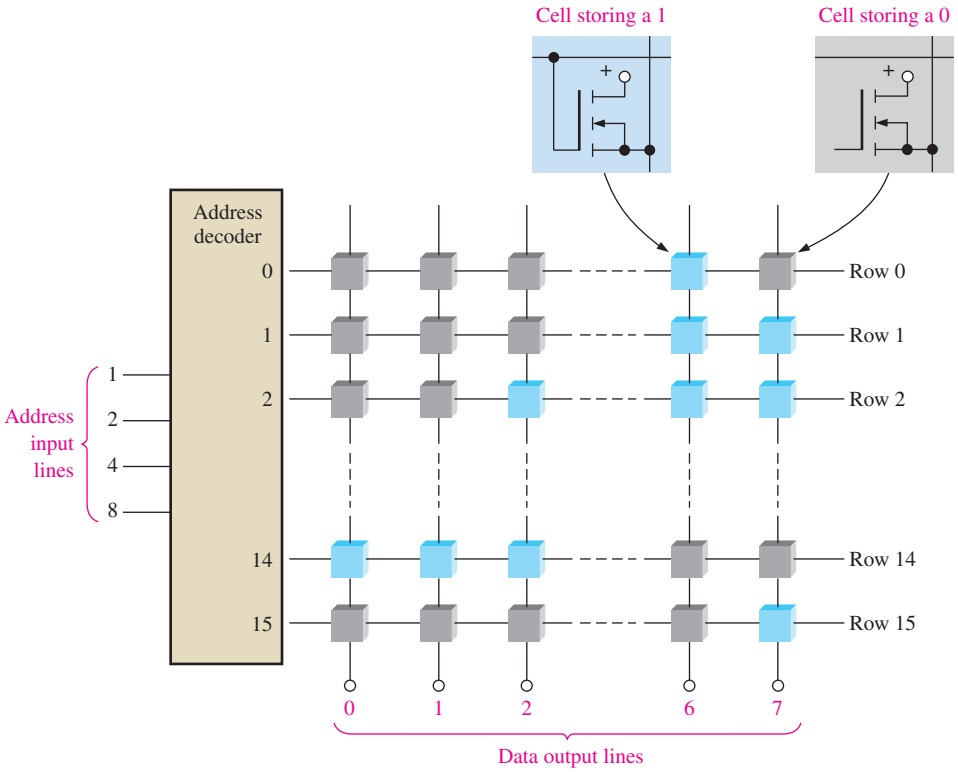


FIGURE 11–25 A representation of a 16 × 8-bit ROM array.

As you can see, the example ROM in Figure 11–25 is organized into 16 addresses, each of which stores 8 data bits. Thus, it is a 16 × 8 (16-by-8) ROM, and its total capacity is 128 bits or 16 bytes. ROMs can be used as look-up tables (LUTs) for code conversions and logic function generation.

EXAMPLE 11–1

Show a basic ROM, similar to the one in Figure 11–25, programmed for a 4-bit binary-to-Gray conversion.

Solution

Review Chapter 2 for the Gray code. Table 11–1 is developed for use in programming the ROM.

The resulting 16 × 4 ROM array is shown in Figure 11–26. You can see that a binary code on the address input lines produces the corresponding Gray code on the output lines (columns). For example, when the binary number 0110 is applied to the address input lines, address 6, which stores the Gray code 0101, is selected.

Related Problem*

Using Figure 11–26, determine the Gray code output when a binary code of 1011 is applied to the address input lines.

*Answers are at the end of the chapter.

TABLE 11–1

Binary				Gray			
B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

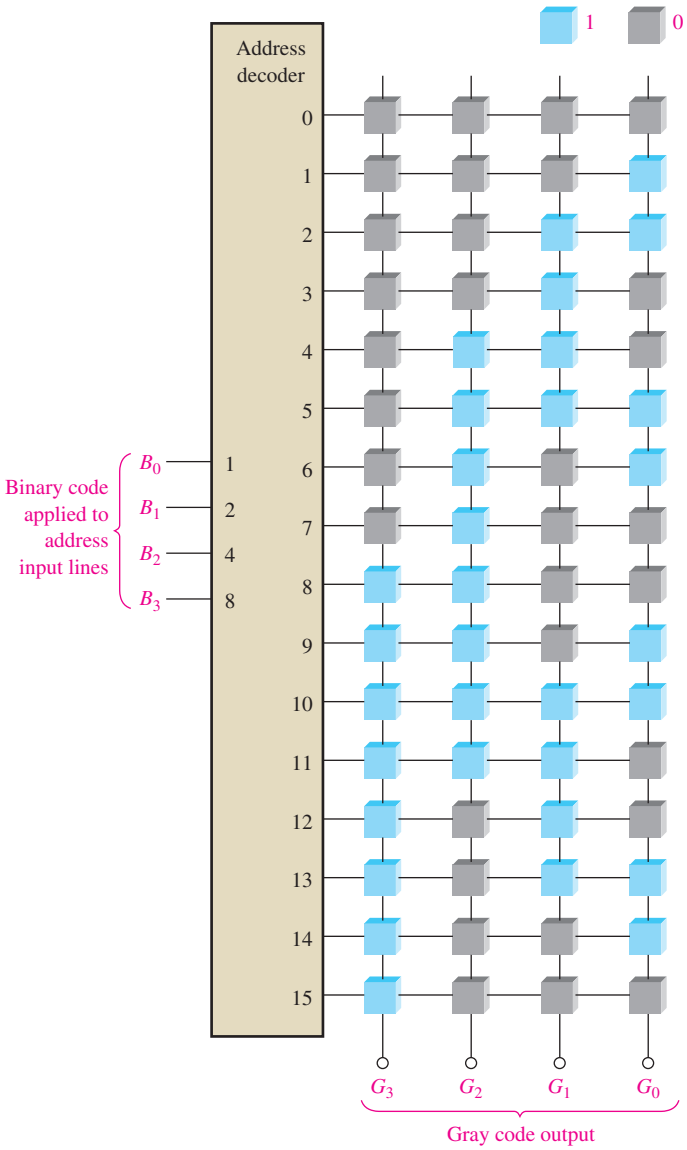


FIGURE 11–26 Representation of a ROM programmed as a binary-to-Gray code converter.

Internal ROM Organization

Most IC ROMs have a more complex internal organization than that in the basic simplified example just presented. To illustrate how an IC ROM is structured, let's use a 1024-bit device with a 256 × 4 organization. The logic symbol is shown in Figure 11–27. When any one of 256 binary codes (eight bits) is applied to the address lines, four data bits appear on the outputs if the chip select inputs are LOW. (256 addresses require eight address lines.)

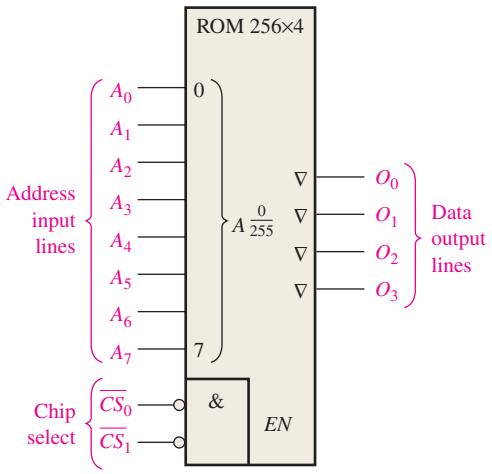


FIGURE 11–27 A 256 × 4 ROM logic symbol. The $A\frac{0}{255}$ designator means that the 8-bit address code selects addresses 0 through 255.

Although the 256 × 4 organization of this device implies that there are 256 rows and 4 columns in the memory array, this is not actually the case. The memory cell array is actually a 32 × 32 matrix (32 rows and 32 columns), as shown in the block diagram in Figure 11–28.

The ROM in Figure 11–28 works as follows. Five of the eight address lines (A_0 through A_4) are decoded by the row decoder (often called the Y decoder) to select one of the 32 rows. Three of the eight address lines (A_5 through A_7) are decoded by the column decoder (often called the X decoder) to select four of the 32 columns. Actually, the column decoder consists of four 1-of-8 decoders (data selectors), as shown in Figure 11–28.

The result of this structure is that when an 8-bit address code (A_0 through A_7) is applied, a 4-bit data word appears on the data outputs when the chip select lines ($\overline{CS}_0$ and $\overline{CS}_1$) are LOW to enable the output buffers. This type of internal organization (architecture) is typical of IC ROMs of various capacities.

InfoNote

ROM is used in a computer to store the BIOS (Basic Input/Output System). These are programs that are used to perform fundamental supervisory and support functions for the computer. For example, BIOS programs stored in the ROM control certain video monitor functions, provide for disk formatting, scan the keyboard for inputs, and control certain printer functions.

ROM Access Time

A typical timing diagram that illustrates ROM access time is shown in Figure 11–29. The **access time,** t_a, of a ROM is the time from the application of a valid address code on the input lines until the appearance of valid output data. Access time can also be measured from the activation of the chip select ($\overline{CS}$) input to the occurrence of valid output data when a valid address is already on the input lines.

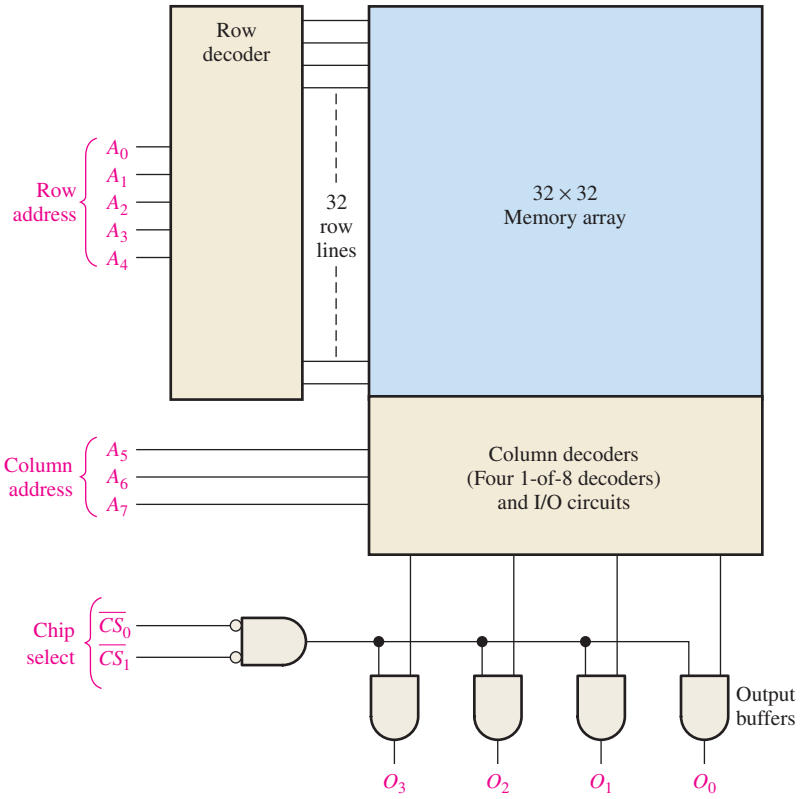


FIGURE 11–28 A 1024-bit ROM with a 256 × 4 organization based on a 32 × 32 array.

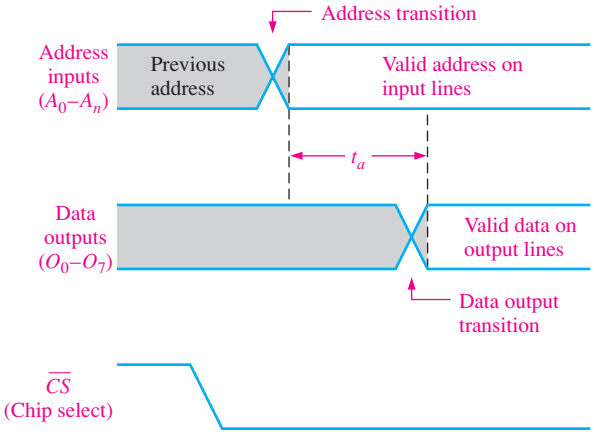


FIGURE 11–29 ROM access time (t_a) from address change to data output with chip select already active.

SECTION 11–3 CHECKUP

1. What is the bit storage capacity of a ROM with a 512 × 8 organization?
2. List the types of read-only memories.
3. How many address bits are required for a 2048-bit memory organized as a 256 × 8 memory?

11–4 Programmable ROMs

Programmable ROMs (PROMs) are basically the same as mask ROMs once they have been programmed. As you have learned, ROMs are a type of programmable logic device. The difference is that PROMs come from the manufacturer unprogrammed and are custom programmed in the field to meet the user's needs.

After completing this section, you should be able to

- Distinguish between a mask ROM and a PROM
- Describe a basic PROM memory cell
- Discuss EPROMs including UV EPROMs and EEPROMs
- Analyze an EPROM programming cycle

PROMs

A **PROM** uses some type of fusing process to store bits, in which a memory *link* is burned open or left intact to represent a 0 or a 1. The fusing process is irreversible; once a PROM is programmed, it cannot be changed.

Figure 11–30 illustrates a MOS PROM array with fusible links. The fusible links are manufactured into the PROM between the source of each cell's transistor and its column line. In the programming process, a sufficient current is injected through the fusible link to burn it open to create a stored 0. The link is left intact for a stored 1.

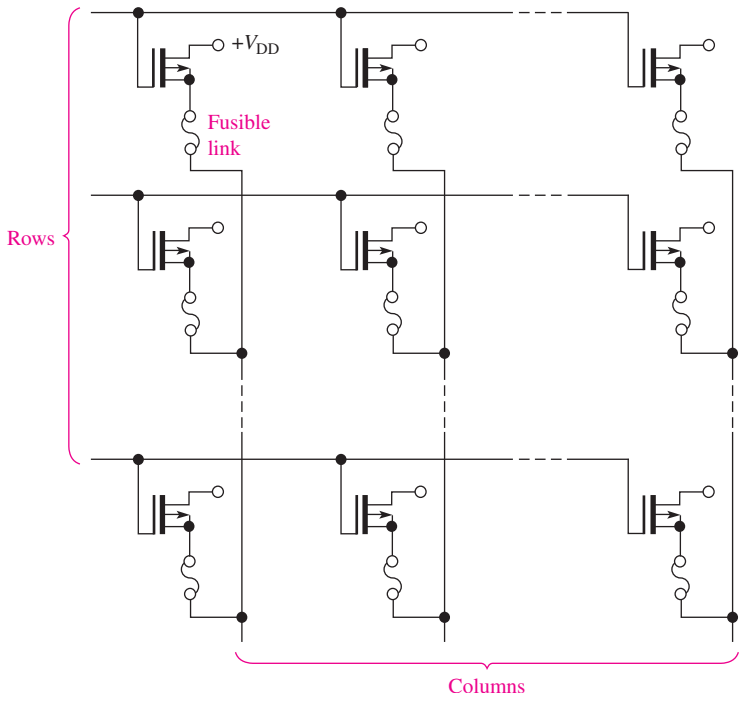


FIGURE 11–30 MOS PROM array with fusible links. (All drains are commonly connected to V_{DD}.)

Three basic fuse technologies used in PROMs are metal links, silicon links, and *pn* junctions. A brief description of each of these follows.

1. Metal links are made of a material such as nichrome. Each bit in the memory array is represented by a separate link. During programming, the link is either "blown" open

or left intact. This is done basically by first addressing a given cell and then forcing a sufficient amount of current through the link to cause it to open.

2. Silicon links are formed by narrow, notched strips of polycrystalline silicon. Programming of these fuses requires melting of the links by passing a sufficient amount of current through them. This amount of current causes a high temperature at the fuse location that oxidizes the silicon and forms an insulation around the now-open link.
3. Shorted junction, or avalanche-induced migration, technology consists basically of two *pn* junctions arranged back-to-back. During programming, one of the diode junctions is avalanched, and the resulting voltage and heat cause aluminum ions to migrate and short the junction. The remaining junction is then used as a forward-biased diode to represent a data bit.

EPROMs

An **EPROM** is an erasable PROM. Unlike an ordinary PROM, an EPROM can be reprogrammed if an existing program in the memory array is erased first.

An EPROM uses an NMOSFET array with an isolated-gate structure. The isolated transistor gate has no electrical connections and can store an electrical charge for indefinite periods of time. The data bits in this type of array are represented by the presence or absence of a stored gate charge. Erasure of a data bit is a process that removes the gate charge.

A typical EPROM is represented in Figure 11–31 by a logic diagram. Its operation is representative of that of other typical EPROMs of various sizes. As the logic symbol shows, this device has 2048 addresses ($2^{11} = 2048$), each with eight bits. Notice that the eight outputs are tri-state (∇).

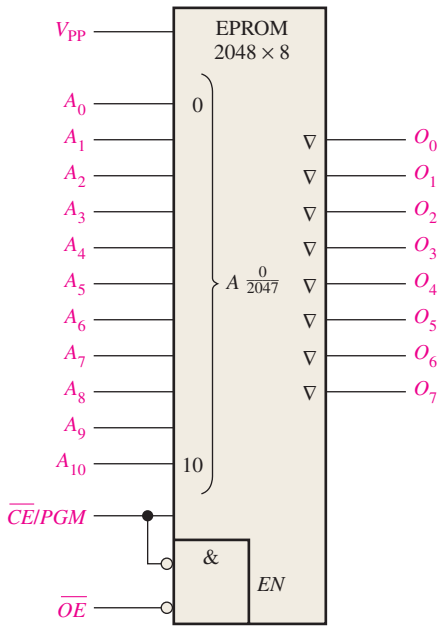


FIGURE 11–31 The logic symbol for a 2048 × 8 EPROM.

To read from the memory, the output enable input ($\overline{OE}$) must be LOW and the power-down/program ($\overline{CE}/PGM$) input LOW.

To program or write to the device, a high dc voltage is applied to V_{PP} and $\overline{OE}$ is HIGH. The eight data bits to be programmed into a given address are applied to the outputs (O_0

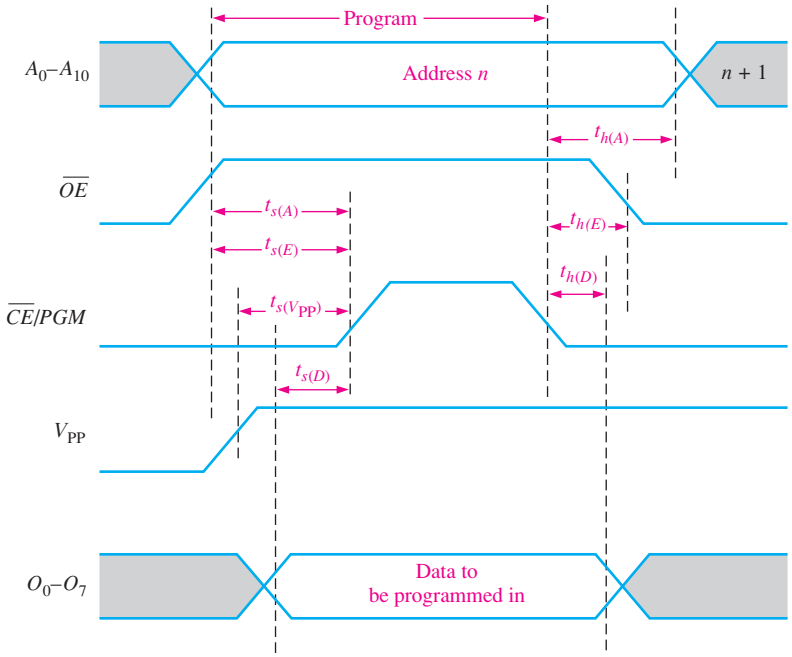


FIGURE 11–32 Timing diagram for a 2048 × 8 EPROM programming cycle, with critical setup times (t_s) and hold times (t_h) indicated.

through O_7), and the address is selected on inputs A_0 through A_{10}. Next, a HIGH level pulse is applied to the $\overline{CE}/PGM$ input. The addresses can be programmed in any order. A timing diagram for the programming is shown in Figure 11–32. These signals are normally produced by an EPROM programmer.

Two basic types of erasable PROMs are, the electrically erasable PROM (EEPROM) and the ultraviolet erasable PROM (UV EPROM). The UV EPROM is much less used than the EEPROM.

EEPROMs

An electrically erasable PROM can be both erased and programmed with electrical pulses. Since it can be both electrically written into and electrically erased, the EEPROM can be rapidly programmed and erased in-circuit for reprogramming. Two types of EEPROMs are the floating-gate MOS and the metal nitride-oxide silicon (MNOS). The application of a voltage on the control gate in the floating-gate structure permits the storage and removal of charge from the floating gate.

UV EPROMs

You can recognize the UV EPROM device by the UV transparent window on the package. The isolated gate in the **FET** of an ultraviolet EPROM is "floating" within an oxide insulating material. The programming process causes electrons to be removed from the floating gate. Erasure is done by exposure of the memory array chip to high-intensity ultraviolet radiation through the UV window on top of the package. The positive charge stored on the gate is neutralized after several minutes to an hour of exposure time.

SECTION 11–4 CHECKUP

1. How do PROMs differ from ROMs?
2. What represents a data bit in an EPROM?
3. What is the normal mode of operation for a PROM?

11–5 The Flash Memory

The ideal memory has high storage capacity, nonvolatility, in-system read and write capability, comparatively fast operation, and cost effectiveness. The traditional memory technologies such as ROM, PROM, EPROM, EEPROM, SRAM, and DRAM individually exhibit one or more of these characteristics. Flash memory has all of the desired characteristics.

After completing this section, you should be able to

- Discuss the basic characteristics of a flash memory
- Describe the basic operation of a flash memory cell
- Compare flash memories with other types of memories
- Discuss the USB flash drive

Flash memories are high-density read/write memories (high-density translates into large bit storage capacity) that are nonvolatile, which means that data can be stored indefinitely without power. High-density means that a large number of cells can be packed into a given surface area on a chip; that is, the higher the density, the more bits that can be stored on a given size chip. This high density is achieved in flash memories with a storage cell that consists of a single floating-gate MOS transistor. A data bit is stored as charge or the absence of charge on the floating gate depending if a 0 or a 1 is stored.

Flash Memory Cell

A single-transistor cell in a flash memory is represented in Figure 11–33. The stacked gate MOS transistor consists of a control gate and a floating gate in addition to the drain and source. The floating gate stores electrons (charge) as a result of a sufficient voltage applied to the control gate. A *0 is stored when there is more charge* and a *1 is stored when there is less or no charge.* The amount of charge present on the floating gate determines if the transistor will turn on and conduct current from the drain to the source when a control voltage is applied during a read operation.

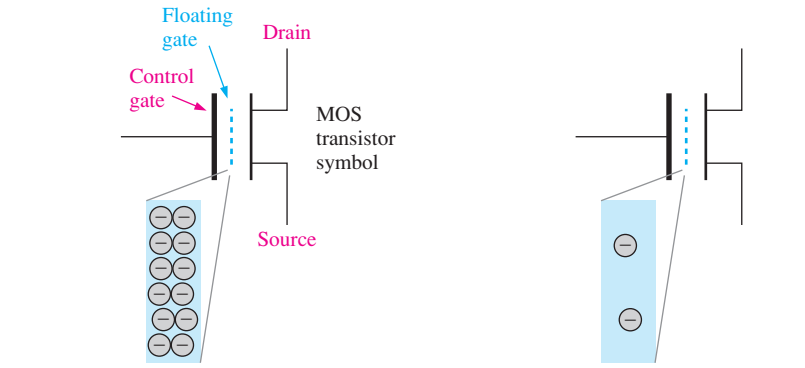


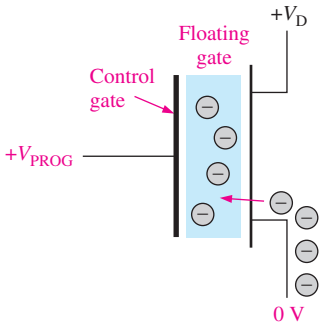
FIGURE 11–33 The storage cell in a flash memory.

Basic Flash Memory Operation

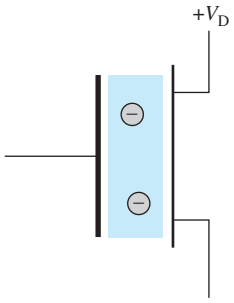
There are three major operations in a flash memory: the *programming* operation, the *read* operation, and the *erase* operation.

Programming

Initially, all cells are at the 1 state because charge was removed from each cell in a previous erase operation. The programming operation adds electrons (charge) to the floating gate of those cells that are to store a 0. No charge is added to those cells that are to store a 1. Application of a sufficient positive voltage to the control gate with respect to the source during programming attracts electrons to the floating gate, as indicated in Figure 11–34. Once programmed, a cell can retain the charge for up to 100 years without any external power.



To store a 0, a sufficient positive voltage is applied to the control gate with respect to the source to add charge to the floating gate during programming.



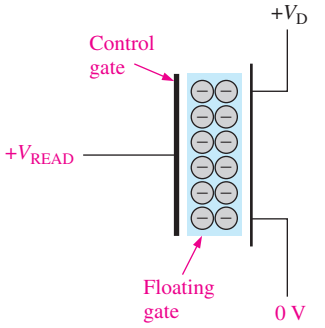
To store a 1, no charge is added and the cell is left in the erased condition.

FIGURE 11–34 Simplified illustration of storing a 0 or a 1 in a flash cell during the programming operation.

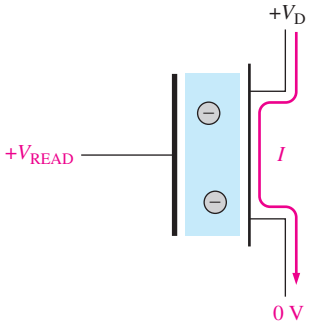
Read

During a read operation, a positive voltage is applied to the control gate. The amount of charge present on the floating gate of a cell determines whether or not the voltage applied to the control gate will turn on the transistor. If a 1 is stored, the control gate voltage is sufficient to turn the transistor on. If a 0 is stored, the transistor will not turn on because the control gate voltage is not sufficient to overcome the negative charge stored in the floating gate. Think of the charge on the floating gate as a voltage source that opposes the voltage applied to the control gate during a read operation. So the floating gate charge associated with a stored 0 prevents the control gate voltage from reaching the turn-on threshold, whereas the small or zero charge associated with a stored 1 allows the control gate voltage to exceed the turn-on threshold.

When the transistor turns on, there is current from the drain to the source of the cell transistor. The presence of this current is sensed to indicate a 1, and the absence of this current is sensed to indicate a 0. This basic idea is illustrated in Figure 11–35.



When a 0 is read, the transistor remains off because the charge on the floating gate prevents the read voltage from exceeding the turn-on threshold.

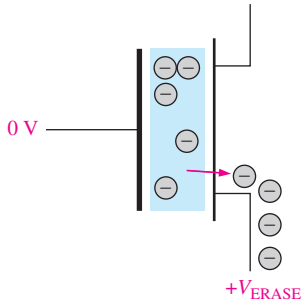


When a 1 is read, the transistor turns on because the absence of charge on the floating gate allows the read voltage to exceed the turn-on threshold.

FIGURE 11–35 The read operation of a flash cell in an array.

Erase

During an erase operation, charge is removed from all the memory cells. A sufficient positive voltage is applied to the transistor source with respect to the control gate. This is opposite in polarity to that used in programming. This voltage attracts electrons from the floating gate and depletes it of charge, as illustrated in Figure 11–36. A flash memory is always erased prior to being reprogrammed.



To erase a cell, a sufficient positive voltage is applied to the source with respect to the control gate to remove charge from the floating gate during the erase operation.

FIGURE 11–36 Simplified illustration of removing charge from a cell during erase.

Flash Memory Array

A simplified array of flash memory cells is shown in Figure 11–37. Only one row line is accessed at a time. When a cell in a given bit line turns on (stored 1) during a read operation, there is current through the bit line, which produces a voltage drop across the active load. This voltage drop is compared to a reference voltage with a comparator circuit and an output level indicating a 1 is produced. If a 0 is stored, then there is no current or little current in the bit line and an opposite level is produced on the comparator output.

The memory stick is a storage medium that uses flash memory technology in a physical configuration smaller than a stick of chewing gum. Memory sticks are typically available up to 64 GB capacities and as a kit with a PC card adaptor. Because of its compact design, it is ideal for use in small digital electronics products, such as laptop computers and digital cameras.

Comparison of Flash Memories with Other Memories

Let's compare flash memories with other types of memories with which you are already familiar.

Flash vs. ROM, EPROM, and EEPROM

Read-only memories are high-density, nonvolatile devices. However, once programmed the contents of a ROM can never be altered. Also, the initial programming is a time-consuming and costly process. The EEPROM has a more complex cell structure than either the ROM or UV EPROM and so the density is not as high, although it can be reprogrammed without being removed from the system. Because of its lower density, the cost/bit is higher than ROMs or EPROMs. Although the UV EPROM is a high-density, nonvolatile memory, it can be erased only by removing it from the system and using ultraviolet light. It can be reprogrammed only with specialized equipment.

A flash memory can be reprogrammed easily in the system because it is essentially a READ/WRITE device. The density of a flash memory compares with the ROM and EPROM because both have single-transistor cells. A flash memory (like a ROM, EPROM, or EEPROM) is nonvolatile, which allows data to be stored indefinitely with power off.

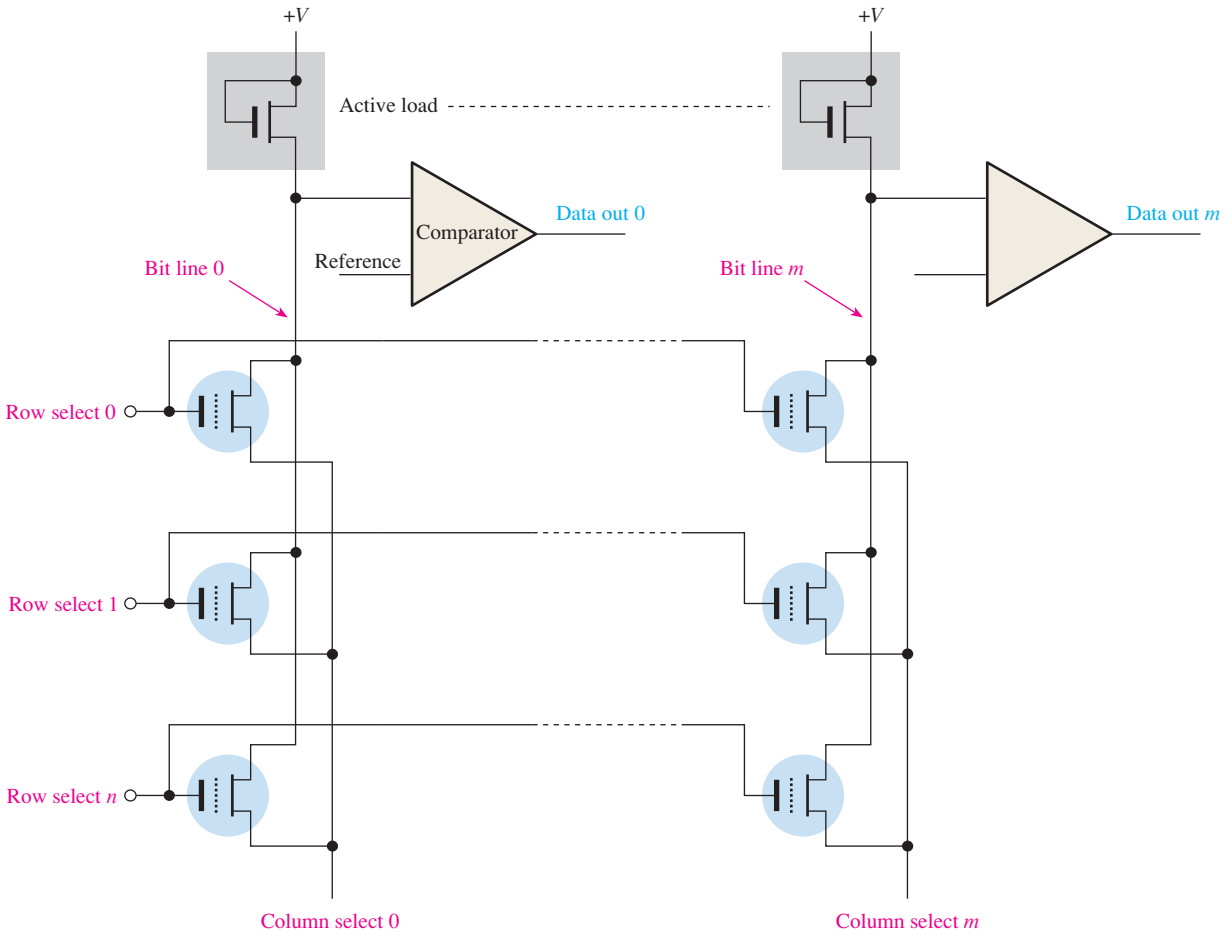


FIGURE 11–37 Basic flash memory array.

Flash vs. SRAM

As you have learned, static random-access memories are volatile READ/WRITE devices. A SRAM requires constant power to retain the stored data. In many applications, a battery backup is used to prevent data loss if the main power source is turned off. However, since battery failure is always a possibility, indefinite retention of the stored data in a SRAM cannot be guaranteed. Because the memory cell in a SRAM is basically a flip-flop consisting of several transistors, the density is relatively low.

A flash memory is also a READ/WRITE memory, but unlike the SRAM it is nonvolatile. Also, a flash memory has a much higher density than a SRAM.

Flash vs. DRAM

Dynamic random-access memories are volatile high-density READ/WRITE devices. DRAMs require not only constant power to retain data but also that the stored data must be refreshed frequently. In many applications, backup storage such as hard disk must be used with a DRAM.

Flash memories exhibit higher densities than DRAMs because a flash memory cell consists of one transistor and does not need refreshing, whereas a DRAM cell is one transistor plus a capacitor that has to be refreshed. Typically, a flash memory consumes much less power than an equivalent DRAM and can be used as a hard disk replacement in many applications.

Table 11–2 provides a comparison of the memory technologies.

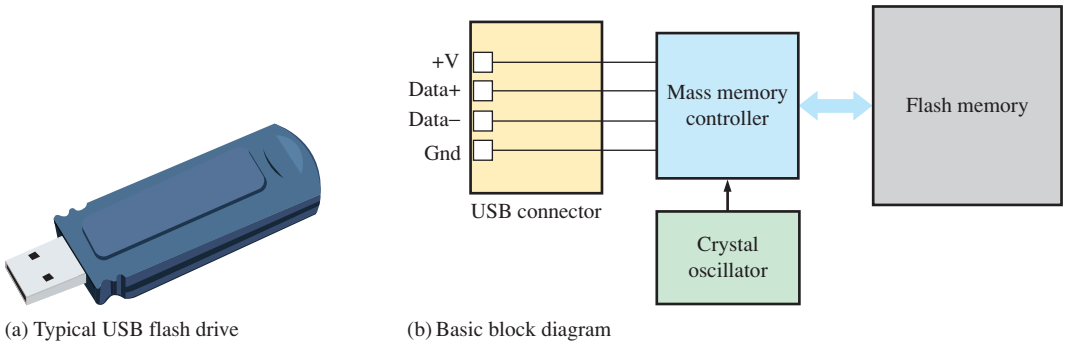
TABLE 11–2

Comparison of types of memories.

Memory Type	Nonvolatile	High-Density	One-Transistor Cell	In-System Writability
Flash	Yes	Yes	Yes	Yes
SRAM	No	No	No	Yes
DRAM	No	Yes	Yes	Yes
ROM	Yes	Yes	Yes	No
EEPROM	Yes	No	No	Yes
UV EPROM	Yes	Yes	Yes	No

USB Flash Drive

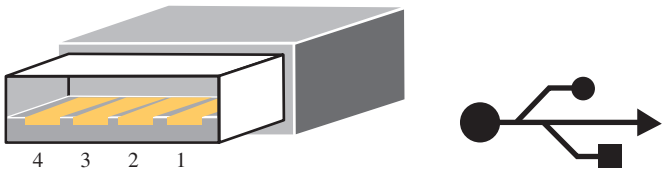
A USB flash drive consists of a flash memory connected to a standard USB connector housed in a small case about the size of a cigarette lighter. The USB connector can be plugged into a port on a personal computer and obtains power from the computer. These memories are usually rewritable and can have a storage capacity up to 512 GB (a number which is constantly increasing), with most ranging from 2 GB to 64 GB. A typical USB flash drive is shown in Figure 11–38(a), and a basic block diagram is shown in part (b).



(a) Typical USB flash drive (b) Basic block diagram

FIGURE 11–38 The USB flash drive.

The USB flash drive uses a standard USB A-type connector for connection to the computer, as shown in Figure 11–39(a). Peripherals such as printers use the USB B-type connector, which has a different shape and physical pin configuration. The USB icon is shown in part (b).



(a) Type A USB connector (b) USB icon

FIGURE 11–39 Connector and symbol.

SECTION 11–5 CHECKUP

1. What types of memories are nonvolatile?
2. What is a major advantage of a flash memory over a SRAM or DRAM?
3. List the three modes of operation of a flash memory.

11–6 Memory Expansion

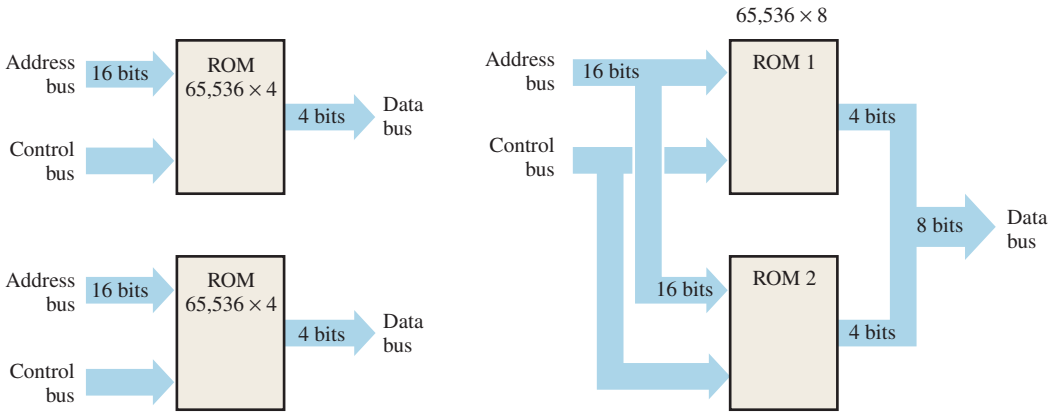
Available memory can be expanded to increase the word length (number of bits in each address) or the word capacity (number of different addresses) or both. Memory expansion is accomplished by adding an appropriate number of memory chips to the address, data, and control buses. SIMMs and DIMMs, which are types of memory expansion modules, are introduced.

After completing this section, you should be able to

- Define *word-length expansion*
- Show how to expand the word length of a memory
- Define *word-capacity expansion*
- Show how to expand the word capacity of a memory
- Discuss types of memory modules

Word-Length Expansion

To increase the **word length** of a memory, the number of bits in the data bus must be increased. For example, an 8-bit word length can be achieved by using two memories, each with 4-bit words as illustrated in Figure 11–40(a). As you can see in part (b), the 16-bit address bus is commonly connected to both memories so that the combination memory still has the same number of addresses ($2^{16} = 65{,}536$) as each individual memory. The 4-bit data buses from the two memories are combined to form an 8-bit data bus. Now when an address is selected, eight bits are produced on the data bus—four from each memory. Example 11–2 shows the details of 65,536 × 4 to 65,536 × 8 expansion.



(a) Two separate 65,536 × 4 ROMs

(b) One 65,536 × 8 ROM from two 65,536 × 4 ROMs

FIGURE 11–40 Expansion of two 65,536 × 4 ROMs into a 65,536 × 8 ROM to illustrate word-length expansion.

EXAMPLE 11–2

Expand the 65,536 × 4 ROM (64k × 4) in Figure 11–41 to form a 64k × 8 ROM. Note that "64k" is the accepted shorthand for 65,536. Why not "65k"? Maybe it's because 64 is also a power-of-two.

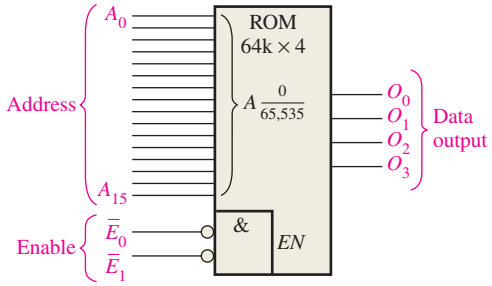


FIGURE 11–41 A 64k × 4 ROM.

Solution

Two 64k × 4 ROMs are connected as shown in Figure 11–42. Notice that a specific address is accessed in ROM 1 and ROM 2 at the same time. The four bits from a selected address in ROM 1 and the four bits from the corresponding address in ROM 2 go out in parallel to form an 8-bit word on the data bus. Also notice that a LOW on the enable line, $\overline{E}$, which forms a simple control bus, enables *both* memories.

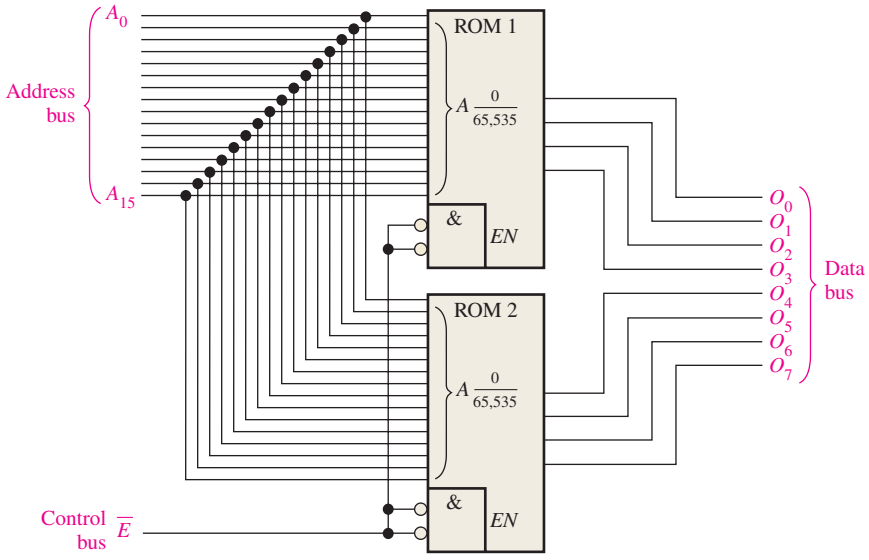


FIGURE 11–42

Related Problem

Describe how you would expand a 64k × 1 ROM to a 64k × 8 ROM.

EXAMPLE 11–3

Use the memories in Example 11–2 to form a 64k × 16 ROM.

Solution

In this case you need a memory that stores 65,536 16-bit words. Four 64k × 4 ROMs are required to do the job, as shown in Figure 11–43.

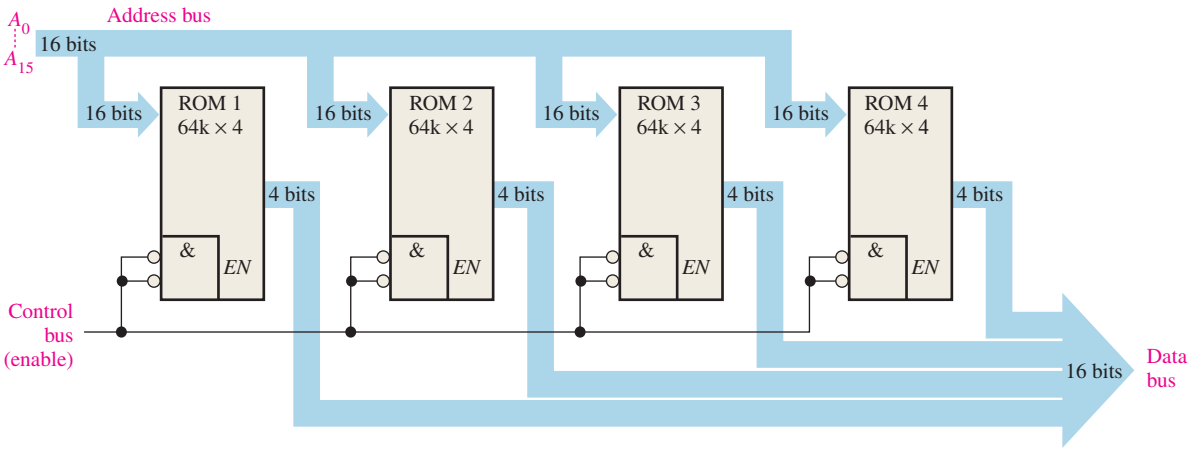


FIGURE 11–43

Related Problem

How many 64k × 1 ROMs would be required to implement the memory shown in Figure 11–43?

A ROM has only data outputs, but a RAM has both data inputs and data outputs. For word-length expansion in a RAM (SRAM or DRAM), the data inputs *and* data outputs form the data bus. Because the same lines are used for data input and data output, tri-state buffers are required. Most RAMs provide internal tri-state circuitry. Figure 11–44 illustrates RAM expansion to increase the data word length.

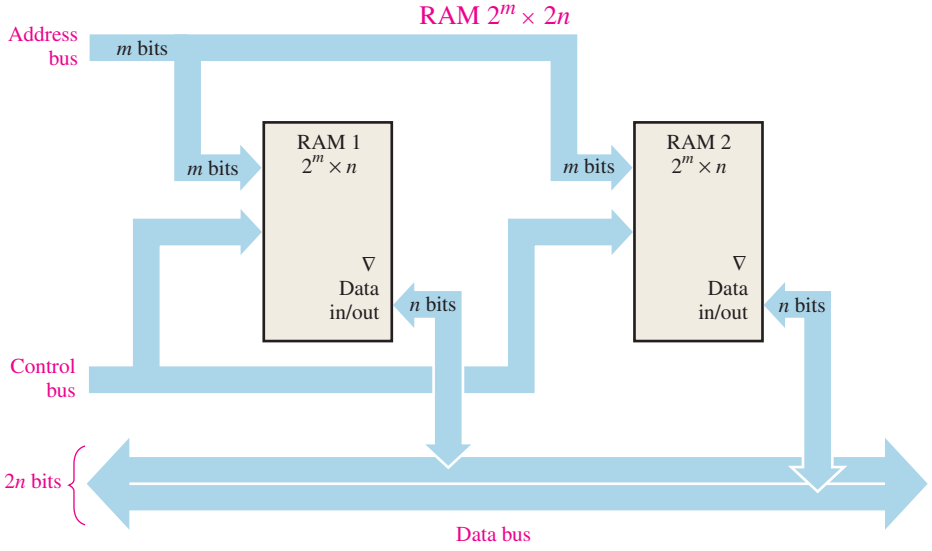


FIGURE 11–44 Illustration of word-length expansion with two $2^m \times n$ RAMs forming a $2^m \times 2n$ RAM.

EXAMPLE 11–4

Use 1M × 4 SRAMs to create a 1M × 8 SRAM.

Solution

Two 1M × 4 SRAMs are connected as shown in the simplified block diagram of Figure 11–45.

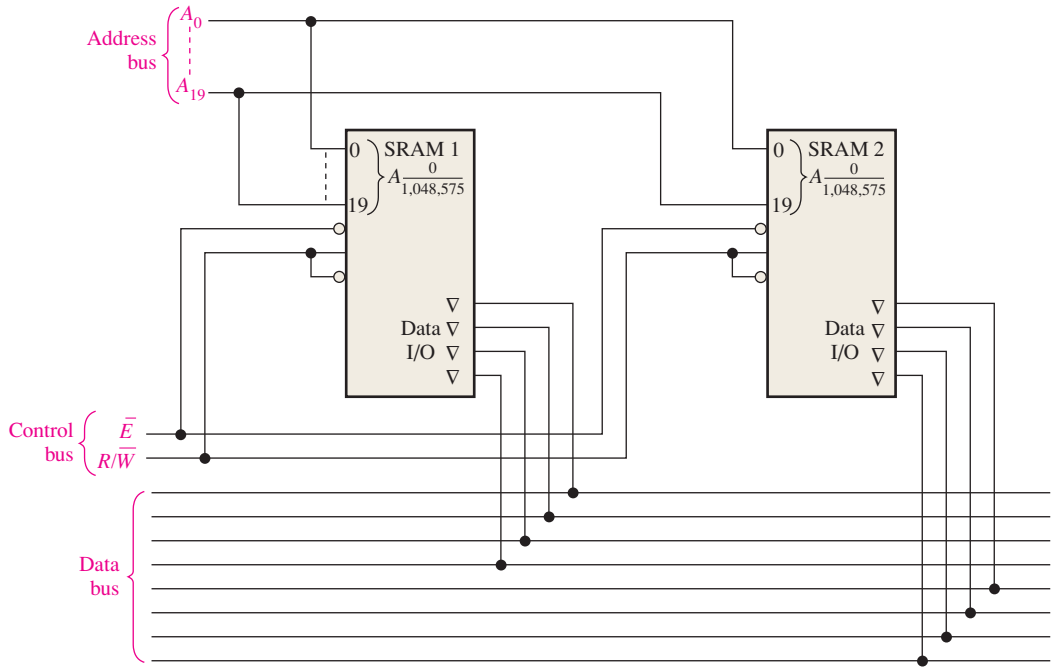


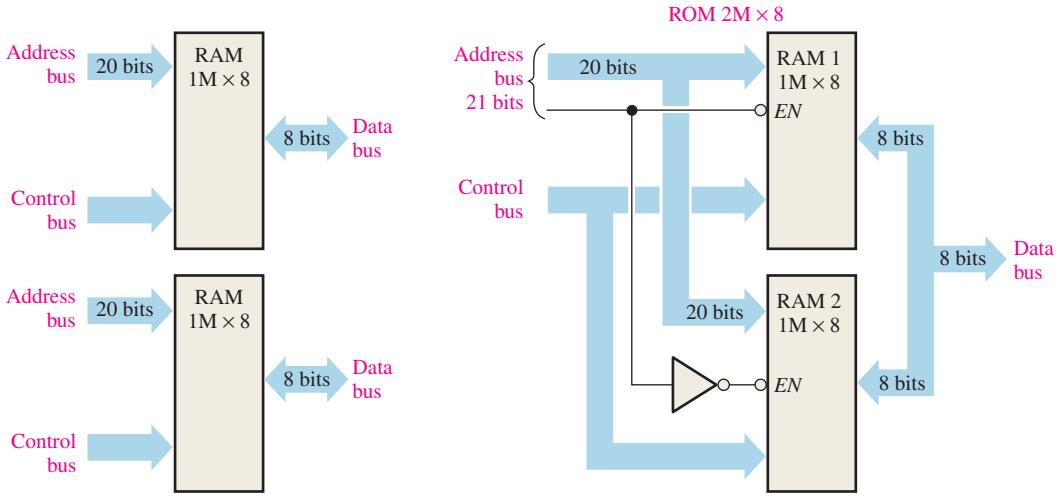
FIGURE 11–45

Related Problem

Use 1M × 8 SRAMs to create a 1M × 16 SRAM.

Word-Capacity Expansion

When memories are expanded to increase the **word capacity**, the *number of addresses is increased.* To achieve this increase, the number of address bits must be increased, as illustrated in Figure 11–46, (where two 1M × 8 RAMs are expanded to form a 2M × 8 memory).



(a) Individual memories each store 1,048,576 8-bit words

(b) Memories expanded to form a 2M × 8 RAM requiring a 21-bit address bus

FIGURE 11–46 Illustration of word-capacity expansion.

Each individual memory has 20 address bits to select its 1,048,576 addresses, as shown in part (a). The expanded memory has 2,097,152 addresses and therefore requires 21 address bits, as shown in part (b). The twenty-first address bit is used to enable the appropriate memory chip. The data bus for the expanded memory remains eight bits wide. Details of this expansion are illustrated in Example 11–5.

EXAMPLE 11–5

Use 512k × 4 RAMs to implement a 1M × 4 memory.

Solution

The expanded addressing is achieved by connecting the enable ($\overline{E}_0$) input to the twentieth address bit (A_{19}), as shown in Figure 11–47. Input $\overline{E}_1$ is used as an enable input common to both memories. When the twentieth address bit (A_{19}) is LOW, RAM 1 is selected (RAM 2 is disabled), and the nineteen lower-order address bits (A_0–A_{18}) access each of the addresses in RAM 1. When the twentieth address bit (A_{19}) is HIGH, RAM 2 is enabled by a LOW on the inverter output (RAM 1 is disabled), and the nineteen lower-order address bits (A_0 – A_{18}) access each of the RAM 2 addresses.

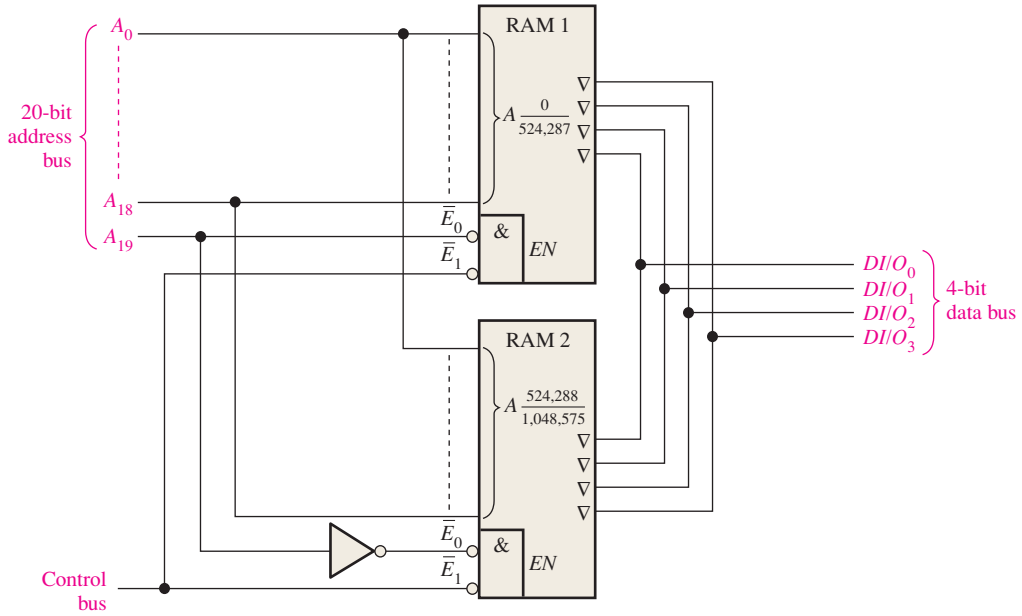


FIGURE 11–47

Related Problem

What are the ranges of addresses in RAM 1 and in RAM 2 in Figure 11–47?

Memory Modules

SDRAMs are available in modules consisting of multiple memory ICs arranged on a printed circuit board (PCB). The most common type of SDRAM memory module is called a **DIMM** (dual in-line memory module). Another version of the DIMM is the SODIMM (small-outline DIMM). A type of memory module, generally found in older equipment and essentially obsolete, is the **SIMM** (single in-line memory module). The SIMM has connection pins on one side of a PCB where the DIMM uses both sides of the board. DIMMs plug into a socket on the system mother board for memory expansion. A generic representation of a memory module is shown in Figure 11–48 with the system board connectors into which the modules are inserted.

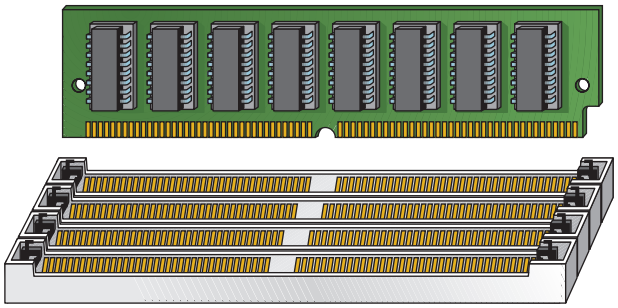


FIGURE 11–48 A memory module with connectors.

DIMMs generally contain DDR SDRAM memory chips. DDR means double data rate, so a DDR SDRAM transfers two blocks of data for each clock cycle rather than one like a standard SDRAM. Three basic types of modules are DDR, DDR2, and DDR3.

- DDR modules have 184 pins and require a 2.5 voltage source.
- DDR2 modules have 240 pins and require a 1.8 voltage source.
- DDR3 modules have 240 pins and require a 1.5 voltage source.

The DDR, DDR2, and DDR3 have transfer data rates of 1600 MB/s, 3200 MB/s, and 6400 MB/s respectively.

Memory components are extremely sensitive to static electricity. Use the following precautions when handling memory chips or modules such as DIMMs:

- Before handling, discharge your body's static charge by touching a grounded surface or wear a grounding wrist strap containing a high-value resistor if available. A convenient, reliable ground is the ac outlet ground.
- Do not remove components from their antistatic bags until you are ready to install them.
- Do not lay parts on the antistatic bags because only the inside is antistatic.
- When handling DIMMs, hold by the edges or the metal mounting bracket. Do not touch components on the boards or the edge connector pins.
- Never slide any part over any type of surface.
- Avoid plastic, vinyl, styrofoam, and nylon in the work area.

When installing DIMMs, follow these steps:

1. Line up the notches on the DIMM board with the notches in the memory socket.
2. Push firmly on the module until it is securely seated in the socket.
3. Generally, the latches on both sides of the socket will snap into place when the module is completely inserted. These latches also release the module, so it can be removed from the socket.

SECTION 11–6 CHECKUP

1. How many 16k × 1 RAMs are required to achieve a memory with a word capacity of 16k and a word length of eight bits?
2. To expand the 16k × 8 memory in question 1 to a 32k × 8 organization, how many more 16k × 1 RAMs are required?
3. What does DIMM stand for?

11–7 Special Types of Memories

In this section, the first in–first out (FIFO) memory, the last in–first out (LIFO) memory, the memory stack, and the charge-coupled device memory are covered.

After completing this section, you should be able to

- Describe a FIFO memory
- Describe a LIFO memory
- Discuss memory stacks
- Explain how to use a portion of RAM as a memory stack
- Describe a basic CCD memory

First In–First Out (FIFO) Memories

This type of memory is formed by an arrangement of shift registers. The term **FIFO** refers to the basic operation of this type of memory, in which the first data bit written into the memory is the first to be read out.

One important difference between a conventional shift register and a FIFO register is illustrated in Figure 11–49. In a conventional register, a data bit moves through the register only as new data bits are entered; in a FIFO register, a data bit immediately goes through the register to the right-most bit location that is empty.

Conventional Shift Register

Input	X	X	X	X	Output
0	0	*X*	*X*	*X*	→
1	1	0	*X*	*X*	→
1	1	1	0	*X*	→
0	0	1	1	1	→

X = unknown data bits.
In a conventional shift register, data stay to the left until "forced" through by additional data.

FIFO Shift Register

Input	—	—	—	—	Output
0	—	—	—	0	→
1	—	—	1	0	→
1	—	1	1	0	→
0	0	1	1	0	→

— = empty positions.
In a FIFO shift register, data "fall" through (go right).

FIGURE 11–49 Comparison of conventional and FIFO register operation.

Figure 11–50 is a block diagram of a FIFO serial memory. This particular memory has four serial 64-bit data registers and a 64-bit control register (marker register). When data are entered by a shift-in pulse, they move automatically under control of the marker register to the empty location closest to the output. Data cannot advance into occupied positions. However, when a data bit is shifted out by a shift-out pulse, the data bits remaining in the registers automatically move to the next position toward the output. In an asynchronous FIFO, data are shifted out independent of data entry, with the use of two separate clocks.

FIFO Applications

One important application area for the FIFO register is the case in which two systems of differing data rates must communicate. Data can be entered into a FIFO register at one rate and taken out at another rate. Figure 11–51 illustrates how a FIFO register might be used in these situations.

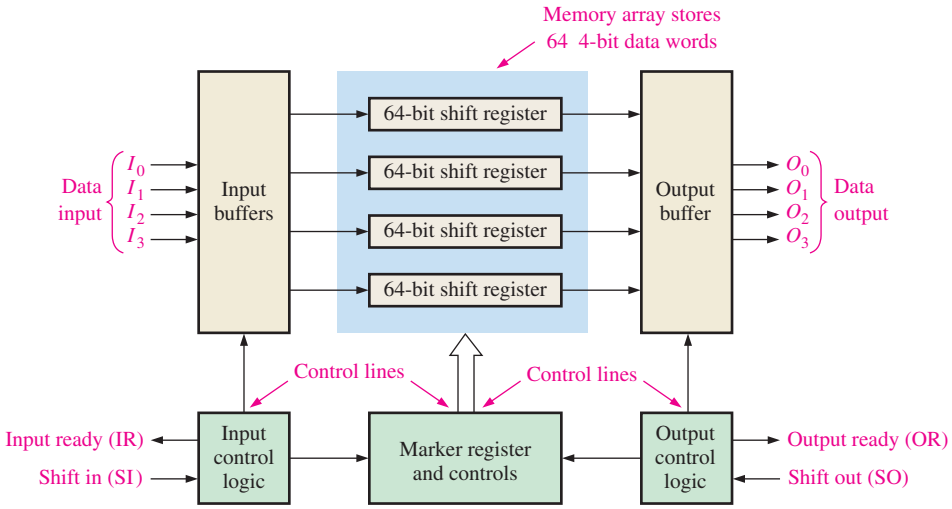


FIGURE 11–50 Block diagram of a typical FIFO serial memory.

(a) Irregular telemetry data can be stored and retransmitted at a constant rate.

(b) Data input at a slow keyboard rate can be stored and then transferred at a higher rate for processing.

(c) Data input at a constant rate can be stored and then output in bursts.

(d) Data in bursts can be stored and reformatted into a constant-rate output.

FIGURE 11–51 Examples of the FIFO register in data-rate buffering applications.

Last In–First Out (LIFO) Memories

The **LIFO** (last in–first out) memory is found in applications involving microprocessors and other computing systems. It allows data to be stored and then recalled in reverse order; that is, the last data byte to be stored is the first data byte to be retrieved.

Register Stacks

A LIFO memory is commonly referred to as a push-down stack. In some systems, it is implemented with a group of registers as shown in Figure 11–52. A stack can consist of any number of registers, but the register at the top is called the *top-of-stack*.

To illustrate the principle, a byte of data is loaded in parallel onto the top of the stack. Each successive byte pushes the previous one down into the next register. This process is illustrated in Figure 11–53. Notice that the new data byte is always loaded into the top register and the previously stored bytes are pushed deeper into the stack. The name *push-down stack* comes from this characteristic.

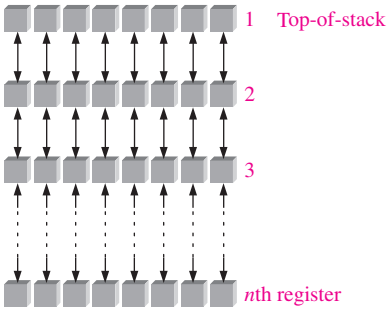


FIGURE 11–52 Register stack.

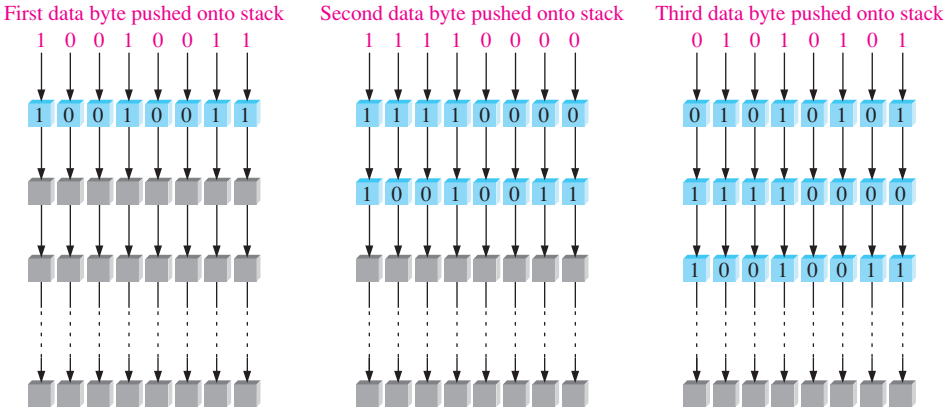


FIGURE 11–53 Simplified illustration of pushing data onto the stack.

Data bytes are retrieved in the reverse order. The last byte entered is always at the top of the stack, so when it is pulled from the stack, the other bytes pop up into the next higher locations. This process is illustrated in Figure 11–54.

RAM Stack

Another approach to LIFO memory used in microprocessor-based systems is the allocation of a section of RAM as the stack rather than the use of a dedicated set of registers. As you have seen, for a register stack the data move up or down from one location to the next. In

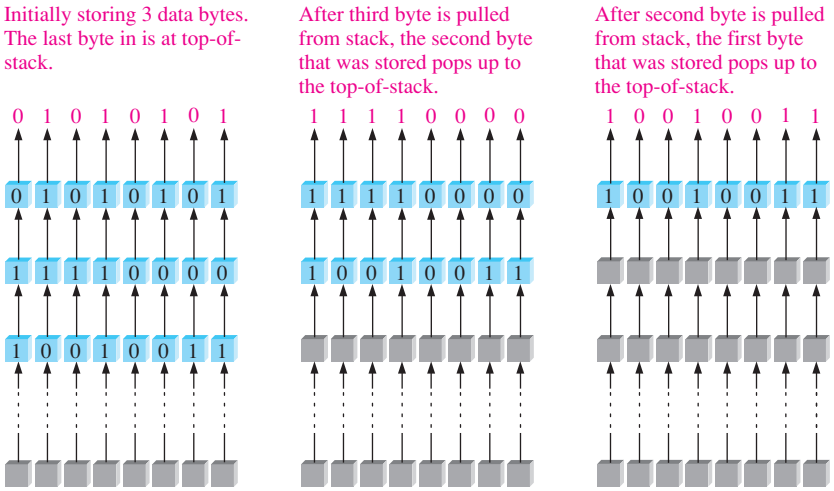


FIGURE 11–54 Simplified illustration of pulling data from the stack.

a RAM stack, the data do not move but the top-of-stack moves under control of a register called the stack pointer.

Consider a random-access memory that is byte organized—that is, one in which each address contains eight bits—as illustrated in Figure 11–55. The binary address 0000000000001111, for example, can be written as 000F in hexadecimal. A 16-bit address can have a *minimum* hexadecimal value of 0000_{16} and a *maximum* value of $FFFF_{16}$. With this notation, a 64 kB memory array can be represented as shown in Figure 11–55. The lowest memory address is 0000_{16} and the highest memory address is $FFFF_{16}$.

Now, consider a section of RAM set aside for use as a stack. A special separate register, the stack pointer, contains the address of the top of the stack, as illustrated in Figure 11–56. A 4-digit hexadecimal representation is used for the binary addresses. In the figure, the addresses are chosen for purposes of illustration.

Now let's see how data are pushed onto the stack. The stack pointer is initially at address $FFEE_{16}$, which is the top of the stack as shown in Figure 11–56(a). The stack pointer is then decremented (decreased) by two to $FFEC_{16}$. This moves the top of the stack to a lower memory address, as shown in Figure 11–56(b). Notice that the top of the stack is not stationary as in the fixed register stack but moves downward (to lower addresses) in the RAM as data words are stored. Figure 11–56(b) shows that two bytes (one data word) are then pushed onto the stack. After the data word is stored, the top of the stack is at $FFEC_{16}$.

Figure 11–57 illustrates the POP operation for the RAM stack. The last data word stored in the stack is read first. The stack pointer that is at FFEC is incremented (increased) by two to address $FFEE_{16}$ and a POP operation is performed as shown in part (b). Keep in mind that RAMs are nondestructive when read, so the data word still remains in the memory after a POP operation. A data word is destroyed only when a new word is written over it.

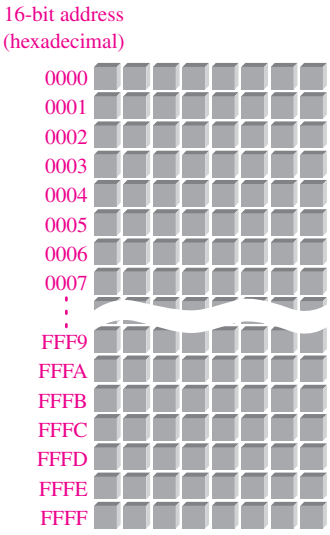
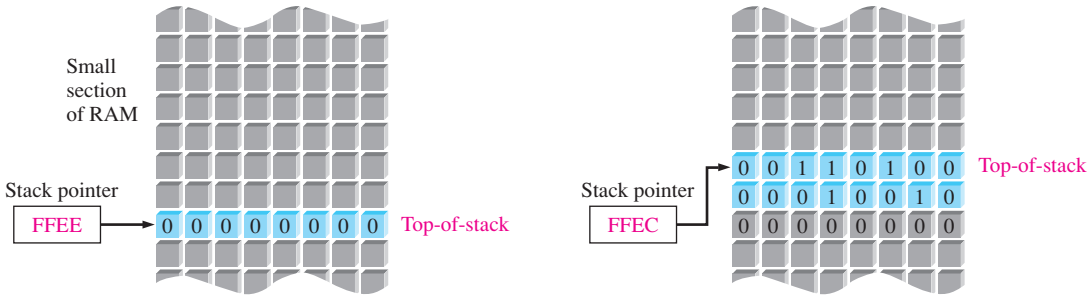


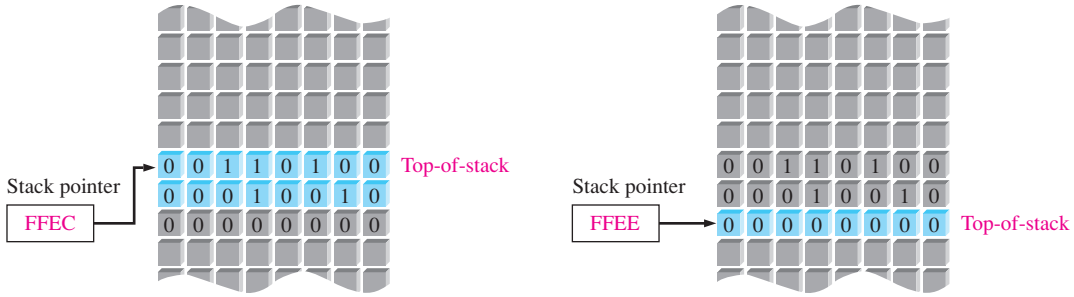
FIGURE 11–55 Representation of a 64 kB memory with the 16-bit addresses expressed in hexadecimal.



(a) The stack pointer is initially at FFEE before the data word 0001001000110100 (1234) is pushed onto the stack.

(b) The stack pointer is decremented by two and the data word 0001001000110100 is placed in the two locations prior to the original stack pointer location.

FIGURE 11–56 Illustration of the PUSH operation for a RAM stack.



(a) The stack pointer is at FFEC before the data word is copied (popped) from the stack.

(b) The stack pointer is incremented by two and the last data word stored is copied (popped) from the stack.

FIGURE 11–57 Illustration of the POP operation for the RAM stack.

A RAM stack can be of any depth, depending on the number of continuous memory addresses assigned for that purpose.

CCD Memories

The **CCD** (charge-coupled device) memory stores data as charges on capacitors and has the ability to convert optical images to electrical signals. Unlike the DRAM, however, the storage cell does not include a transistor. High density is the main advantage of CCDs, and these devices are widely used in digital imaging.

The CCD memory consists of long rows of semiconductor capacitors, called *channels.* Data are entered into a channel serially by depositing a small charge for a 0 and a large charge for a 1 on the capacitors. These charge packets are then shifted along the channel by clock signals as more data are entered.

As with the DRAM, the charges must be refreshed periodically. This process is done by shifting the charge packets serially through a refresh circuit. Figure 11–58 shows the basic concept of a CCD channel. Because data are shifted serially through the channels, the CCD memory has a relatively long access time. CCD arrays are used in many modern cameras to capture video images in the form of light-induced charge.

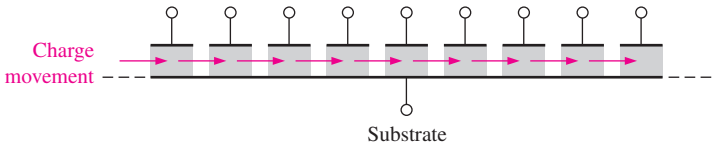


FIGURE 11–58 A CCD (charge-coupled device) channel.

SECTION 11–7 CHECKUP

1. What is a FIFO memory?
2. What is a LIFO memory?
3. Explain the PUSH operation in a memory stack.
4. Explain the POP operation in a memory stack.
5. What does the term *CCD* stand for?

11–8 Magnetic and Optical Storage

In this section, the basics of magnetic disks, magnetic tape, magneto-optical disks, and optical disks are introduced. These storage media are important, particularly in computer applications, where they are used for mass nonvolatile storage of data and programs.

After completing this section, you should be able to

- Describe a magnetic hard disk
- Discuss magnetic tape
- Discuss removable hard disks
- Explain the principle of magneto-optical disks
- Discuss the CD-ROM, CD-R, and CD-RW disks
- Describe the WORM
- Discuss the DVD-ROM

Magnetic Storage

Magnetic Hard Disks

Computers use hard disks as the internal mass storage media. **Hard disks** are rigid "platters" made of aluminum alloy or a mixture of glass and ceramic covered with a magnetic coating. Hard disk drives mainly come in three diameter sizes, 3.5 in., 2.5 in., and 1.8 in. Older formats of 8 in. and 5.25 in. are considered obsolete. A hard disk drive is hermetically sealed to keep the disks dust-free.

Typically, two or more disks are stacked on top of each other on a common shaft or spindle that turns the assembly at several thousand rpm. A separation between each disk allows for a magnetic read/write head that is mounted on the end of an actuator arm, as shown in Figure 11–59. There is a read/write head for both sides of each disk since data are recorded on both sides of the disk surface. The drive actuator arm synchronizes all the read/write heads to keep them in perfect alignment as they "fly" across the disk surface with a separation of only a fraction of a millimeter from the disk. A small dust particle could cause a head to "crash," causing damage to the disk surface.

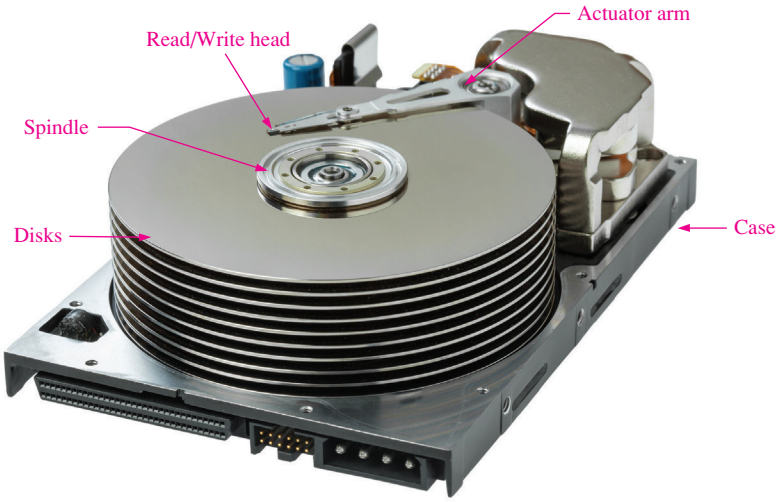


FIGURE 11–59 A hard disk drive. FrameAngel/Shutterstock

Basic Read/Write Head Principles

The hard drive is a random-access device because it can retrieve stored data anywhere on the disk in any order. A simplified diagram of the magnetic surface read/write operation is shown in Figure 11–60. The direction or polarization of the magnetic domains on the disk surface is controlled by the direction of the magnetic flux lines (magnetic field) produced

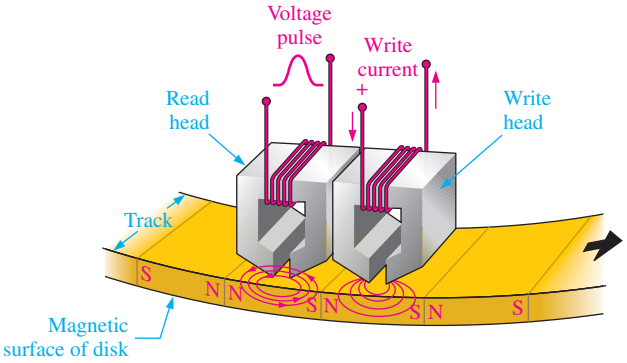


FIGURE 11–60 Simplified read/write head operation.

InfoNote

Data are stored on a hard drive in the form of files. Keeping track of the location of files is the job of the device driver that manages the hard drive (sometimes referred to as hard drive BIOS). The device driver and the computer's operating system can access two tables to keep track of files and file names. The first table is called the FAT (File Allocation Table). The FAT shows what is assigned to specific files and keeps a record of open sectors and bad sectors. The second table is the Root Directory which has file names, type of file, time and date of creation, starting cluster number, and other information about the file.

by the write head according to the direction of a current pulse in the winding. This magnetic flux magnetizes a small spot on the disk surface in the direction of the magnetic field. A magnetized spot of one polarity represents a binary 1, and one of the opposite polarity represents a binary 0. Once a spot on the disk surface is magnetized, it remains until written over with an opposite magnetic field.

When the magnetic surface passes a read head, the magnetized spots produce magnetic fields in the read head, which induce voltage pulses in the winding. The polarity of these pulses depends on the direction of the magnetized spot and indicates whether the stored bit is a 1 or a 0. The read and write heads are usually combined in a single unit.

Hard Disk Format

A hard disk is organized or formatted into tracks and sectors, as shown in Figure 11–61(a). Each track is divided into a number of sectors, and each track and sector has a physical address that is used by the operating system to locate a particular data record. Hard disks typically have from a few hundred to thousands of tracks and are available with storage capacities of up to 1 TB or more. As you can see in the figure, there is a constant number of tracks/sector, with outer sectors using more surface area than the inner sectors. The arrangement of tracks and sectors on a disk is known as the *format*.

A hard disk stack is illustrated in Figure 11–61(b). Hard disk drives differ in the number of disks in a stack, but there is always a minimum of two. All of the same corresponding tracks on each disk are collectively known as a cylinder, as indicated.

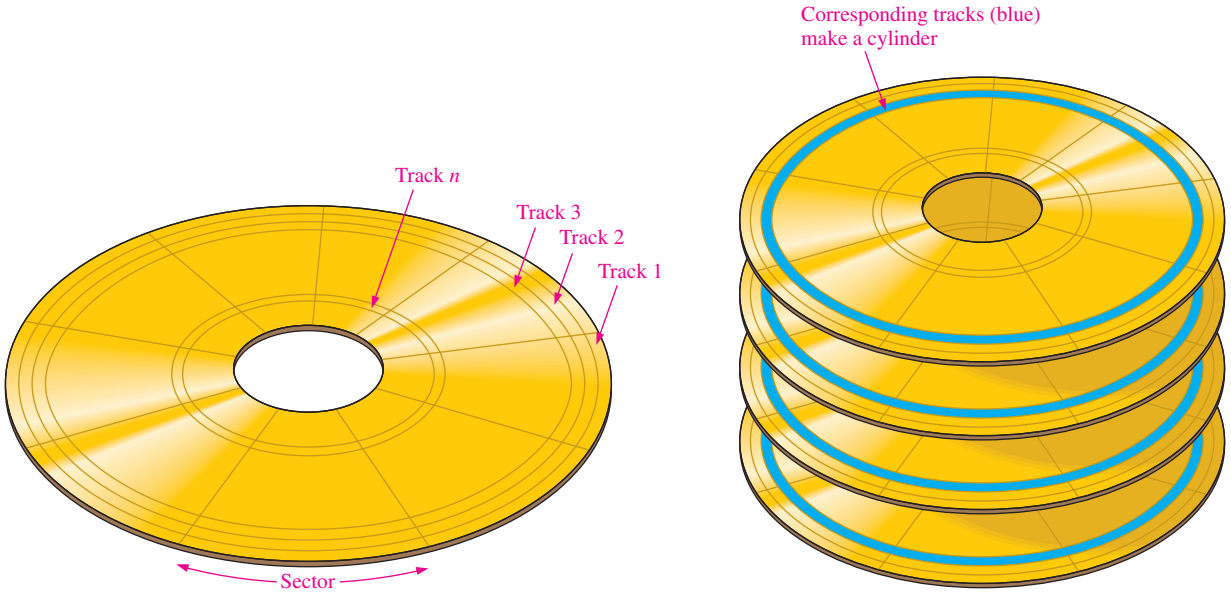


FIGURE 11–61 Hard disk organization and formatting.

Hard Disk Performance

Several basic parameters determine the performance of a given hard disk drive. A *seek* operation is the movement of the read/write head to the desired track. The **seek time** is the average time for this operation to be performed. Typically, hard disk drives have an average seek time of several milliseconds, depending on the particular drive.

The **latency period** is the time it takes for the desired sector to spin under the head once the head is positioned over the desired track. A worst case is when the desired sector is just past the head position and spinning away from it. The sector must rotate almost a full

revolution back to the head position. *Average latency period* assumes that the disk must make half of a revolution. Obviously, the latency period depends on the constant rotational speed of the disk. Disk rotation speeds are different for different disk drives but typically are from 4200 rpm to 15,000 rpm.

The sum of the average seek time and the average latency period is the *access time* for the disk drive.

Removable Hard Disk

A removable hard disk drive with a capacity of 1 TB is available. Keep in mind that the technology is changing so rapidly that there most likely will be further advancements at the time you are reading this.

Magnetic Tape

Tape is used for backup data from mass storage devices and typically is slower than disks because data on tape is accessed serially rather than randomly. There are several types that are available, including QIC, 8 mm, and DLT.

QIC is an abbreviation for quarter-inch cartridge and looks much like audio tape cassettes with two reels inside. Various QIC standards have from 28 to 108 tracks that can store from 80 MB to 1.6 GB. More recent innovations under the Travan standard have lengthened the tape and increased its width allowing storage capacities up to 10 GB. QIC tape drives use read/write heads that have a single write head with a read head on each side. This allows the tape drive to verify data just written when the tape is running in either direction. In the record mode, the tape moves past the read/write heads at approximately 100 inches/second, as indicated in Figure 11–62.

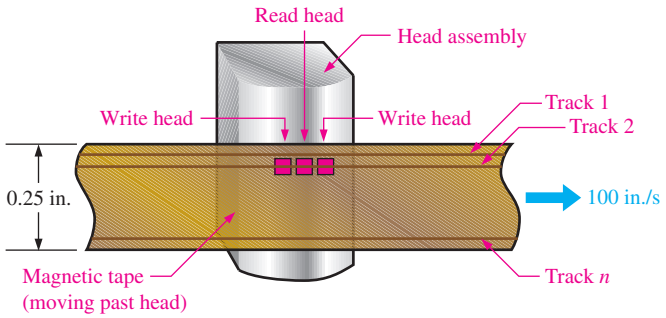


FIGURE 11–62 QIC tape.

8 mm tape was originally designed for the video industry but has been adopted by the computer industry as a reliable way to store large amounts of computer data.

DLT is an abbreviation for digital linear tape. DLT is a half-inch wide tape, which is 60% wider than 8 mm and, of course, twice as wide as standard QIC. Basically, DLT differs in the way the tape-drive mechanism works to minimize tape wear compared to other systems. DLT offers the highest storage capacity of all the tape formats with capacities ranging up to 800 GB.

Magneto-Optical Storage

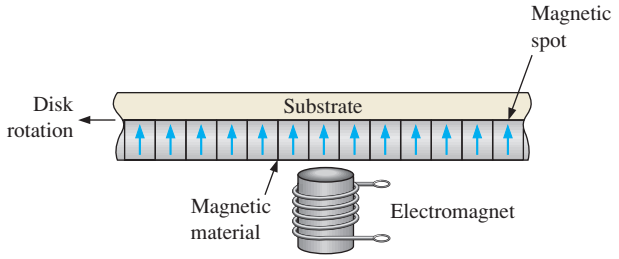
As the name implies, magneto-optical (MO) storage devices use a combination of magnetic and optical (laser) technologies. A **magneto-optical disk** is formatted into tracks and sectors similar to magnetic disks.

The basic difference between a purely magnetic disk and an MO disk is that the magnetic coating used on the MO disk requires heat to alter the magnetic polarization. Therefore, the MO is extremely stable at ambient temperature, making data unchangeable. To write a data bit, a high-power laser beam is focused on a tiny spot on the disk, and the

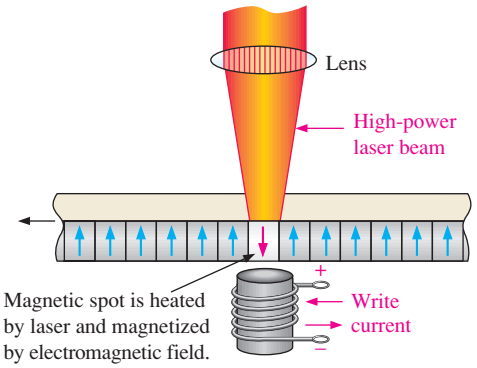
InfoNote

Tape is a viable alternative to disk due to its lower cost per bit. Though the density is lower than for a disk drive, the available surface on a tape is far greater. The highest-capacity tape media are generally on the same order as the largest available disk drive (about 1 TB—a terabyte is one trillion bytes.) Tape has historically offered enough advantage in cost over disk storage to make it a viable product, particularly for backup, where media removability is also important.

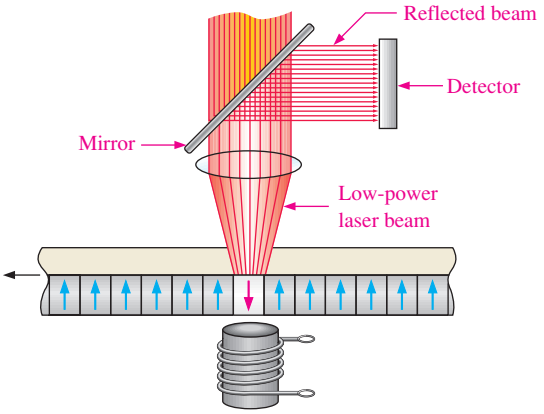
temperature of that tiny spot is raised above a temperature level called the Curie point (about 200°C). Once heated, the magnetic particles at that spot can easily have their direction (polarization) changed by a magnetic field generated by the write head. Information is read from the disk with a less-powerful laser than used for writing, making use of the Kerr effect where the polarity of the reflected laser light is altered depending on the orientation of the magnetic particles. Magnetic spots of one polarity represent 0s and magnetic spots of the opposite polarity represent 1s. Basic MO operation is shown in Figure 11–63, which represents a small cross-sectional area of a disk.



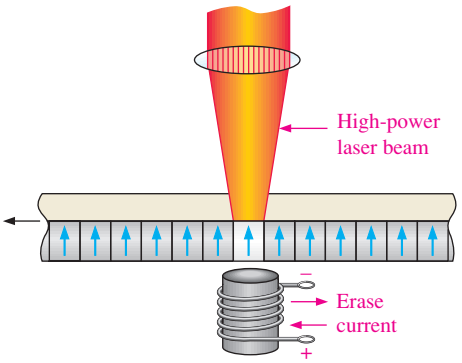
(a) Unrecorded disk



(b) Writing: A high-power laser beam heats the spot, causing the magnetic particles to align with the electromagnetic field.



(c) Reading: A low-power laser beam reflects off of the reversed-polarity magnetic particles and its polarization shifts. If the particles are not reversed, the polarization of the reflected beam is unchanged.



(d) Erasing: The electromagnetic field is reversed as the high-power laser beam heats the spot, causing the magnetic particles to be restored to the original polarity.

FIGURE 11–63 Basic principle of a magneto-optical disk.

Optical Storage

CD-ROM

The most common Compact Disk–Read-Only Memory is a 120 mm diameter disk with a sandwich of three coatings: a polycarbonate plastic on the bottom, a thin aluminum sheet for reflectivity, and a top coating of lacquer for protection. The **CD-ROM** disk is formatted in a single spiral track with sequential 2 kB sectors and has a capacity of 680 MB. Data are prerecorded at the factory in the form of minute indentations called *pits* and the flat area surrounding the pits called *lands*. The pits are stamped into the plastic layer and cannot be erased.

A CD player reads data from the spiral track with a low-power infrared laser, as illustrated in Figure 11–64. The data are in the form of pits and lands as shown. Laser light

reflected from a pit is 180° out-of-phase with the light reflected from the lands. As the disk rotates, the narrow laser beam strikes the series of pits and lands of varying lengths, and a photodiode detects the difference in the reflected light. The result is a series of 1s and 0s corresponding to the configuration of pits and lands along the track.

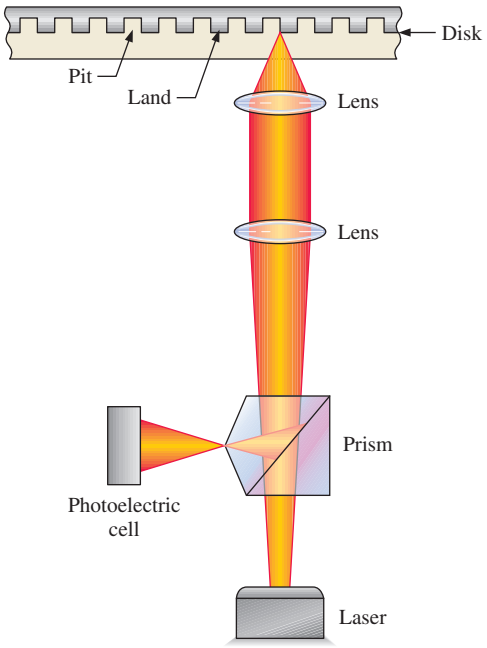


FIGURE 11–64 Basic operation of reading data from a CD-ROM.

WORM

Write Once/Read Many (**WORM**) is a type of optical storage that can be written onto one time after which the data cannot be erased but can be read many times. To write data, a low-power laser is used to burn microscopic pits on the disk surface. 1s and 0s are represented by the burned and nonburned areas.

CD-R

This is essentially a type of WORM. The difference is that the CD-Recordable allows multiple write sessions to different areas of the disk. The **CD-R** disk has a spiral track like the CD-ROM, but instead of mechanically pressing indentations on the disk to represent data, the CD-R uses a laser to burn microscopic spots into an organic dye surface. When heated beyond a critical temperature with a laser during read, the burned spots change color and reflect less light than the nonburned areas. Therefore, 1s and 0s are represented on a CD-R by burned and nonburned areas, whereas on a CD-ROM they are represented by pits and lands. Like the CD-ROM, the data cannot be erased once it is written.

CD-RW

The CD-Rewritable disk can be used to read and write data. Instead of the dye-based recording layer in the CD-R, the **CD-RW** commonly uses a crystalline compound with a special property. When it is heated to a certain temperature, it becomes crystalline when it cools; but if it is heated to a certain higher temperature, it melts and becomes amorphous when it cools. To write data, the focused laser beam heats the material to the melting temperature resulting in an amorphous state. The resulting amorphous areas reflect less light than the crystalline areas, allowing the read operation to detect 1s and 0s. The data can be erased or overwritten by heating the amorphous areas to a temperature above the crystallization

temperature but lower than the melting temperature that causes the amorphous material to revert to a crystalline state.

DVD-ROM

Originally DVD was an abbreviation for Digital Video Disk but eventually came to represent *Digital Versatile Disk.* Like the CD-ROM, **DVD-ROM** data are prestored on the disk. However, the pit size is smaller than for the CD-ROM, allowing more data to be stored on a track. The major difference between CD-ROM and DVD-ROM is that the CD is single-sided, while the DVD has data on both sides. Also, in addition to double-sided DVD disks, there are also multiple-layer disks that use semitransparent data layers placed over the main data layers, providing storage capacities of tens of gigabytes. To access all the layers, the laser beam requires refocusing going from one layer to the other.

Blu-Ray

The **Blu-ray** Disc (BD) is designed to eventually replace the DVD. The BD is the same size as DVDs and CDs. The name *Blu-ray* refers to the blue laser used to read the disc. DVDs use a red laser that has a longer wavelength. Information can be stored on a BD at a greater density and video definition than is possible with a DVD. The smaller Blu-ray laser beam can read recorded data in pits that are less than half the size of the pits on a DVD. A Blu-ray Disc can store about five times more data than a DVD. Typical storage capacities for conventional Blu-ray dual-layer discs are 50 GB, which is the industry standard for feature-length video. Triple layer and quadruple layer discs (BD-XL) can store 100 GB and 128 GB, respectively. Storage capacities up to 1 TB are currently under development.

SECTION 11–8 CHECKUP

1. List the major types of magnetic storage.
2. Generally, how is a magnetic disk organized?
3. How are data written on and read from a magneto-optical disk?
4. List the types of optical storage.

11–9 Memory Hierarchy

A memory system performs the data storage function in a computer. The memory system holds data temporarily during processing and also stores data and programs on a long-term basis. A computer has several types of memory, such as registers, cache, main, and hard disk. Other types of storage can also be used, such as magnetic tape, optical disk, and magnetic disk. Memory hierarchy as well as the system processor determines the processing speed of a computer.

After completing this section, you should be able to

- Discuss several types of memory
- Define memory hierarchy
- Describe key elements in a memory hierarchy

Three key characteristics of memory are cost, capacity, and access time. Memory cost is usually specified in cost per bit. The capacity of a memory is measured in the amount of data (bits or bytes) it can store. The access time is the time it takes to acquire a specified unit of data from the memory. The greater the capacity, the smaller the cost and the greater the access time. The smaller the access time, the greater the cost. The goal of using

a memory hierarchy is to obtain the shortest possible average access time while minimizing the cost.

The speed with which data can be processed depends both on the processor speed and on the time it takes to access stored data. **Memory hierarchy** is the arrangement of various memory elements within the computer architecture to maximize processing speed and minimize cost. Memory can be classified according to its "distance" from the processor in terms of the number of machine cycles or access time required to get data for processing. Distance is measured in time, not in physical location. Faster memory elements are considered closer to the processor compared to slower types of memory elements. Also, the cost per bit is much greater for the memory close to the processor than for the memory that is further from the processor. Figure 11–65 illustrates the arrangement of elements in a typical memory hierarchy.

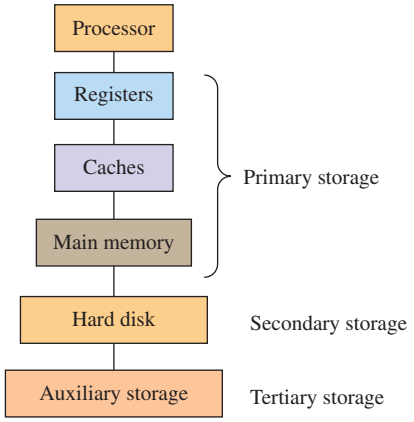


FIGURE 11–65 Typical memory hierarchy.

A primary distinction between the storage elements in Figure 11–65 is the time required for the processor to access data and programs. This access time is known as **memory latency**. The greater the latency, the further from the processor a storage element is considered to be. For example, typical register latency can be up to 1 or 2 ns, cache latency can be up to about 50 ns, main memory latency can be up to about 90 ns, and hard disk latency can be up to about 20 ms. Auxiliary memory latency can range up to several seconds.

Registers

Registers are memory elements that are located within the processor. They have a very small latency as well as a low capacity (number of bits that can be stored). One goal of programming is to keep as much frequently used data in the registers as possible. The number of registers in a processor can vary from the tens to hundreds.

Caches

The next level in the hierarchy is the memory cache, which provides temporary storage. The L1 cache is located in the processor, and the L2 cache is outside of the processor. A programming goal is to keep as much of a program as possible in the cache, especially the parts of a program that are most extensively used. There can be more than two caches in a memory system.

Main Memory

Main memory generally consists of two elements: RAM (random-access memory) and ROM (read-only memory). The RAM is a working memory that temporarily stores less

frequently used data and program instructions. The RAM is volatile, which means that the stored contents are lost when the power is turned off. The ROM is for permanent storage of frequently used programs and data; ROM is nonvolatile. Registers, caches, and main memory are considered primary storage.

Hard Disk

The hard disk has a very high latency and is used for mass storage of data and programs on a permanent basis. The hard disk is also used for virtual memory, space allocated for data when the primary memory fills up. In effect, virtual memory simulates primary memory with the disadvantage of high latency. Capacities range up to about 1 terabyte (TB).

$$1\text{ TB} = 1{,}000{,}000{,}000{,}000\text{ B} = 10^{12}\text{ B}$$

In addition to the internal hard disk, secondary storage can also include off-line storage. Off-line storage includes DVDs, CD-ROM, CD-RW, and USB flash drive. Off-line storage is removable storage.

Auxiliary Storage

Auxiliary storage, also called *tertiary storage,* includes magnetic tape libraries and optical jukeboxes. A **tape library** can store immense amounts of data (up to hundreds of petabytes). A petabyte (PB) is

$$1\text{ PB} = 1{,}000{,}000{,}000{,}000{,}000\text{ B} = 10^{15}\text{ B}$$

An **optical jukebox** is a robotic storage device that automatically loads and unloads optical disks. It may have as many as 2,000 slots for disks and can store hundreds of petabytes.

Relationship of Cost, Capacity, and Access Time

Figure 11–66 shows how capacity (the amount of data a memory can store) and cost per unit of storage varies as the distance from the processor, in terms of access time or latency, increases. The capacity increases and the cost decreases as access time increases.

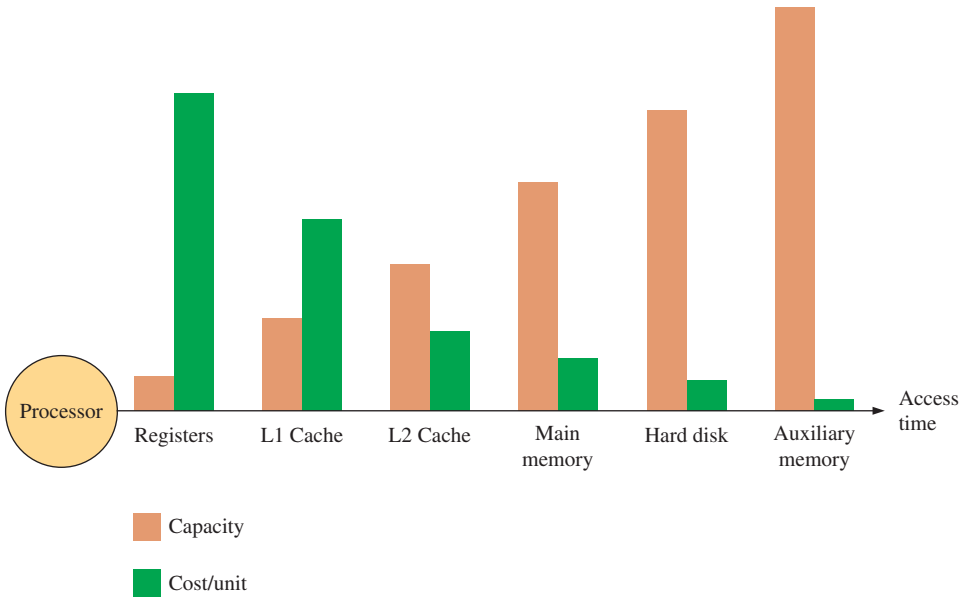


FIGURE 11–66 Changes in memory capacity and cost per unit of data as latency (access time) increases.

Memory Hierarchy Performance

In a computer system, the overall processing speed is usually limited by the memory, not the processor. Programming determines how well a particular memory hierarchy is utilized. The goal is to process data at the fastest rate possible. Two key factors in establishing maximum processor performance are locality and hit rate.

If a block of data is referenced, it will tend to be either referenced again soon or a nearby data block will be referenced soon. Frequent referencing of the same data block is known as *temporal locality*, and the program code should be arranged so that the piece of the data in the cache is reused frequently. Referencing an adjacent data block is known as *spatial locality*, and the program code should be arranged to use consecutive pieces of data on a frequent basis.

A **miss** is a failed attempt by the processor to read or write a block of data in a given level of memory (such as the cache). A miss causes the processor to have to go to a lower level of memory (such as main memory), which has a longer latency. The three types of misses are instruction read miss, data read miss, and data write miss. A successful attempt to read or write a block of data in a given level of memory is called a *hit*. Hits and misses are illustrated in Figure 11–67, where the processor is requesting data from the cache.

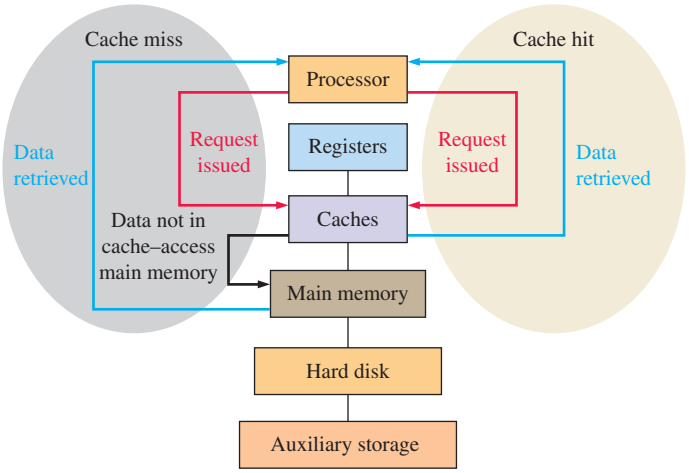


FIGURE 11–67 Illustration of a cache hit and a miss.

The **hit rate** is the percentage of memory accesses that find the requested data in the given level of memory. The *miss rate* is the percentage of memory accesses that fail to find the requested data in the given level of memory and is equal to 1– hit rate. The time required to access the requested information in a given level of memory is called the *hit time*. The higher the hit rate (hit to miss ratio), the more efficient the memory hierarchy is.

SECTION 11–9 CHECKUP

1. State the purpose of memory hierarchy.
2. What is access time?
3. How does memory capacity affect the cost per bit?
4. Does higher level memory generally have lower capacity than lower level memory?
5. What is a hit? A miss?
6. What determines the efficiency of the memory hierarchy?

11–10 Cloud Storage

Cloud storage is a system, usually maintained by a third party, for securely storing data in a remote location that can be conveniently accessed through the Internet. A file on a computer can be stored on secure remote servers and accessed by various user devices such as computers, smart phones, and tablets. Cloud storage eliminates the need for local backup storage such as external hard drives or CDs. When you use cloud storage, you are essentially storing your files or documents on Internet servers instead of or in addition to a computer. The term *cloud* may have originated from the use of a symbol that resembled a cloud on early network diagrams.

After completing this section, you should be able to

- Describe cloud storage
- Explain what a server is
- State the advantages of cloud storage
- Describe several properties of cloud storage

The Cloud Storage System

A **cloud storage** system consists of a remote network of servers (also called *nodes*) that are connected to a user device through the Internet, as shown in Figure 11–68. Some cloud storage systems accommodate only certain types of data such as e-mail or digital pictures, while others store all types of data and range in size from small operations with a few servers to very large operations that utilize hundreds of servers. A facility that houses cloud storage systems is called a **data center**. A typical storage cloud system can serve multiple users.

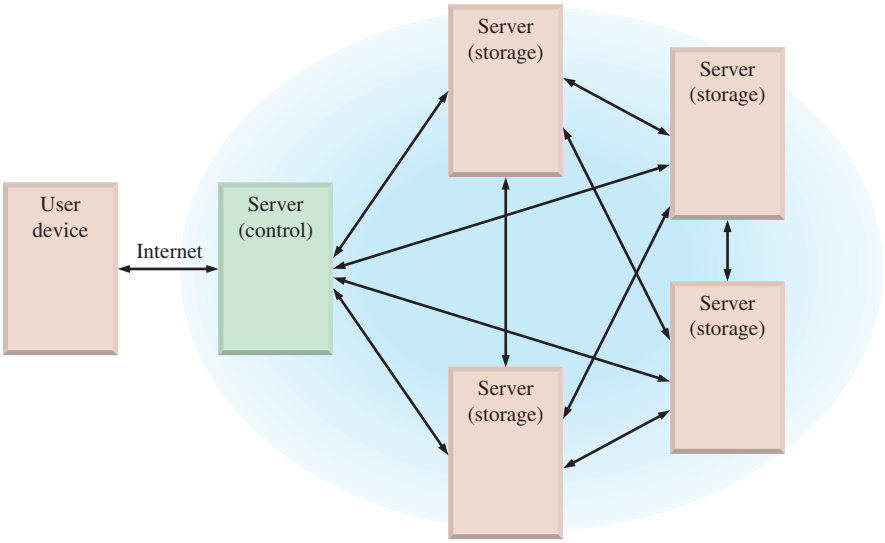


FIGURE 11–68 A typical cloud storage system architecture consists of a master control server and several storage servers that can be accessed by a user device over the Internet.

Servers typically operate within a client-server architecture, where the client is the user that is subscribing to the cloud storage. Theoretically, a **server** is any computerized process that shares a resource with one or more clients. More practically, a storage server is a computer and software with a large memory capacity that responds to requests across a network to provide file storage and access as well as services such as file sharing. The control server

(a) A typical rack of servers (b) A typical server room in a data center

FIGURE 11–69 Cloud servers. (a) Jojje/Shutterstock (b) Oleksiy Mark/Shutterstock

coordinates the activities within the storage cloud network among other servers and manages user access. A server rack and data center are shown in Figure 11–69.

At its simplest level, a cloud storage system needs just one storage server connected to the Internet. When copies of a file are sent by a client to the server over the Internet, the data are stored. When the client wishes to retrieve the data, the storage server (node) sends it back through a Web-based interface or allows the client to manipulate the file on the server itself.

Most cloud storage systems have many storage servers (hundreds in some cases) to provide both capacity and redundancy. A grouping of servers is sometimes called a *cluster*. Depending on the system architecture, a given system may have multiple clusters. A simple system with four storage servers illustrating file storage redundancy is shown in Figure 11–70. When a client sends data to the cloud, it is stored in multiple servers. This redundancy guarantees availability of data at any time to the client and makes the system highly reliable. Redundancy is necessary because a server requires periodic maintenance or may break down and need repairs. In addition to storage server redundancy, most cloud storage systems use power supply redundancy so that all servers are not operating from the same power source.

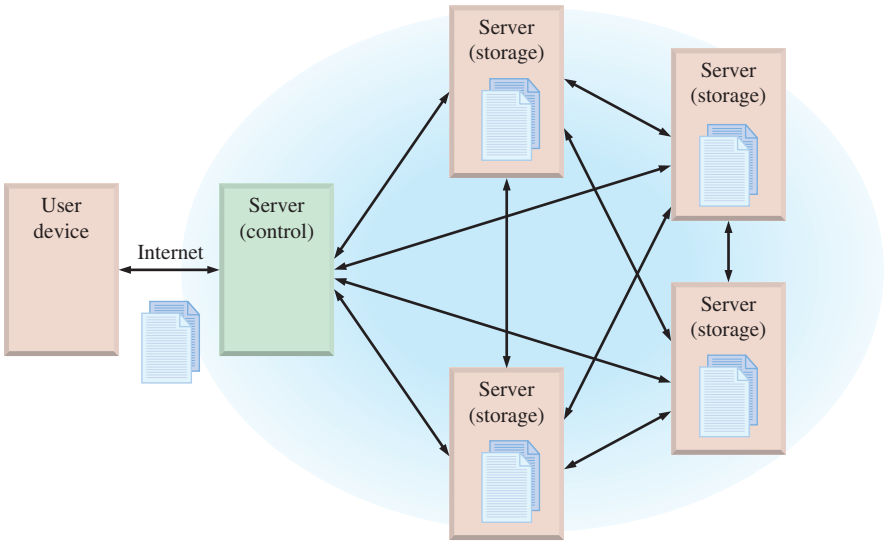


FIGURE 11–70 A simple cloud storage system with storage redundancy. In this case, the data are stored on four different servers.

In addition to reliability that provides assurance that a client's data are accurately stored and can be retrieved at any time, a second major factor for cloud storage is security that the data cannot be compromised. Generally, three methods are used to provide data security:

- *Encryption* or encoding, which prevents the data from being read or interpreted without proper decryption tools
- *Authentication*, which requires a name and password for access
- *Authorization*, which requires a list of only those people who can have access to the data

Cloud storage has certain advantages over traditional data storage in a computer. One advantage is that you can store and retrieve data from any physical location that has Internet access. A second advantage is that you don't have to use the same computer to store and retrieve data or carry a physical storage device for data backup around with you. Also, the user does not have to maintain the storage components. Another advantage of cloud storage is that other people can access your data (data sharing).

Architecture

The term *architecture* relates to how a cloud storage system is structured and organized. The primary purpose of cloud storage architecture is to deliver the service for data storage in a specific way. Architectures vary but generically most consist of a front end, a control, and a back end, as depicted in Figure 11–71.

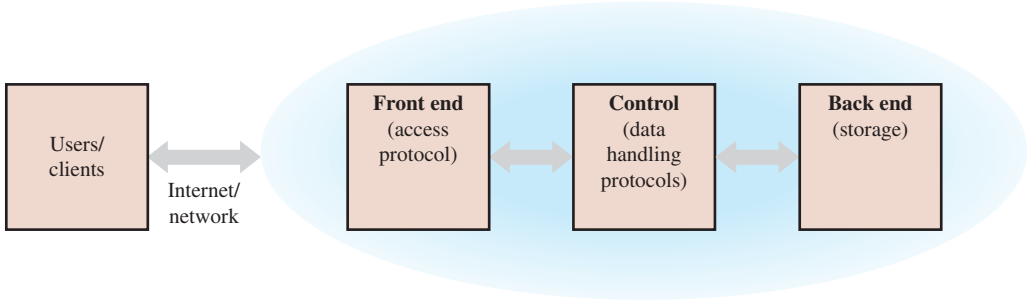


FIGURE 11–71 Generic architecture of a cloud storage system.

A cloud storage system uses various protocols within the architecture that determine how the data are accessed and handled. A **protocol** is a standardized set of software regulations, requirements, and procedures that control and regulate the transmission, processing, and exchange of data among devices. For example, common Internet protocols are HTTP (Hypertext Transfer Protocol), FTP (File Transfer Protocol), TCP/IP (Transfer Control Protocol/Internet Protocol), and SMTP (Simple Mail Transfer Protocol).

An API is an Application Programming Interface, which is essentially a protocol for access and utilization of a cloud storage system. There are many types of APIs. For example, a commonly used one is the REST API. REST stands for Representational State Transfer. An API is a software-to-software interface, not a user interface. With APIs, applications talk to each other "behind the scene" without user knowledge.

Cloud Storage Properties

The following cloud storage properties determine the performance of the system.

- *Latency.* The time between a request for data and the delivery of the data to the user is the **latency** of a system. Delay is due to the time for each component of the cloud storage system to respond to a request and to the time for data to be transferred to the user.

- *Bandwidth.* Bandwidth is a measure of the range of frequencies that can be simultaneously transferred to the cloud and is defined as a range of frequencies that can be handled by the system. Generally, the wider the bandwidth, the shorter the latency and vice versa.
- *Scalability.* The **scalability** property indicates the ability of a cloud storage system to handle increasing amounts of data in a smooth and easy manner; or it is the cloud's ability to improve movement of data through the system (throughput) when additional resources (typically hardware) are added. When the performance of a system improves proportionally to the storage capacity added, the system is said to be scalable. Scaling vertically (scale up) occurs when resources (hardware and memory) are added to a single server (node). Scaling horizontally (scale out) occurs when more servers (nodes) are added to a system.
- *Elasticity.* **Elasticity** is a cloud's ability to deal with variations in the amount of data (load) being transferred in and out of the storage system without service interrupts. There is a subtle difference between scalability and elasticity when describing a system's behavior. Essentially, *scalability* is a static parameter that indicates how much the system can be expanded, and *elasticity* is a dynamic parameter that refers to the implementation of scalability. For example, a storage system may be scalable from one to 100 servers. If the system is currently operating with 20 servers (nodes) and the data load doubles, its elasticity allows 20 more nodes to be added for a total of 40. Likewise, if the data load decreases by half, the elasticity allows 10 nodes to be removed. A server can be added or removed by powering it up or down in a proper manner without disrupting service to the user. Elasticity results in cost efficiency because only the number of servers required for the data load at any given time are consuming power.
- *Multitenancy.* The **multitenancy** property of a cloud storage system allows multiple users to share the same software applications and hardware and the same data storage mechanism but not to see each other's data.

SECTION 11–10 CHECKUP

1. What is a cloud storage system?
2. What is a server?
3. How does a user connect to a cloud storage system?
4. Name three advantages of a cloud system.