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

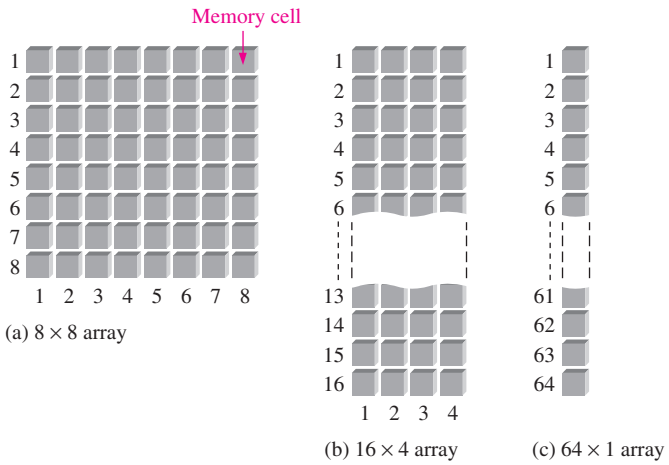
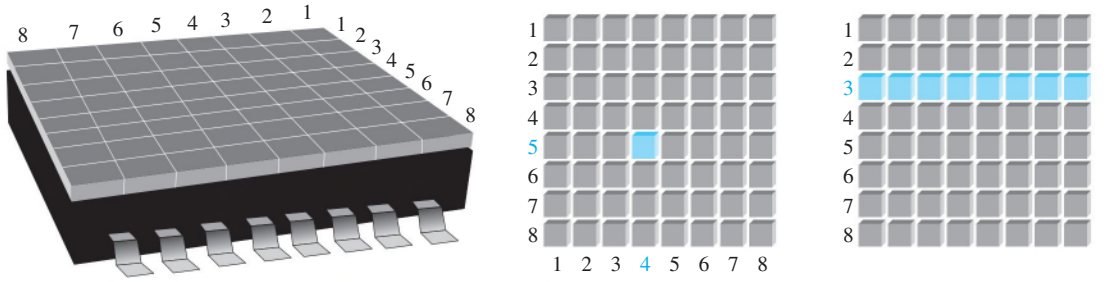


FIGURE 11–1 A 64-cell memory array organized in three different ways.

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.



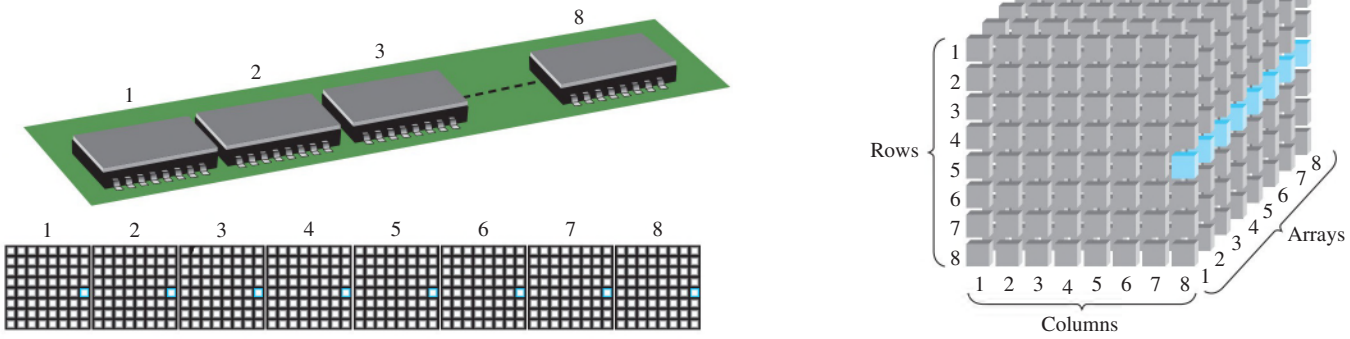
(a) Physical structure of 64-bit memory.

(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.

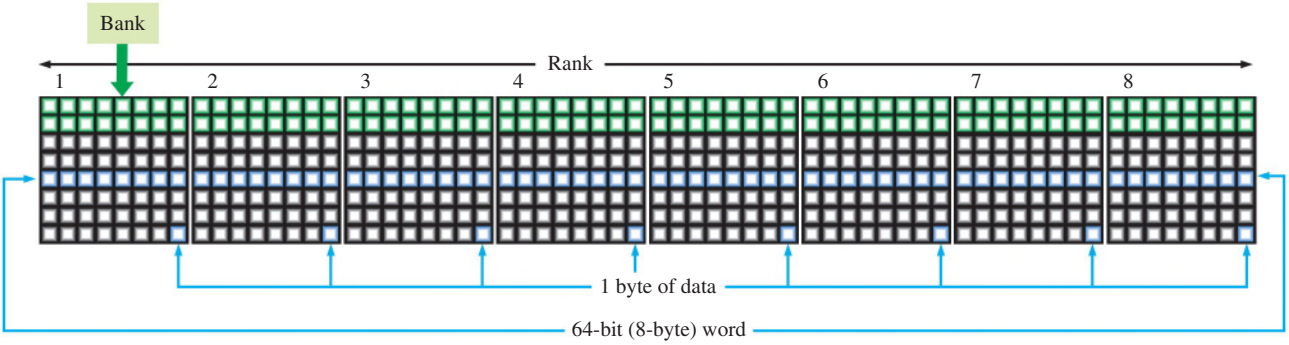


FIGURE 11–4 Simple illustration of memory bank and memory rank.

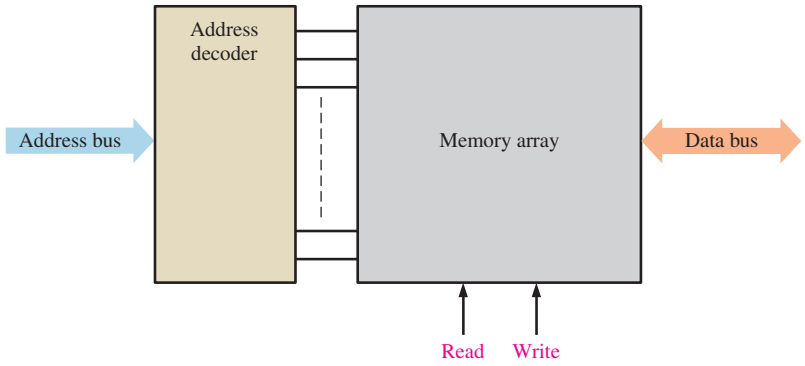
Basic Memory Operations

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.

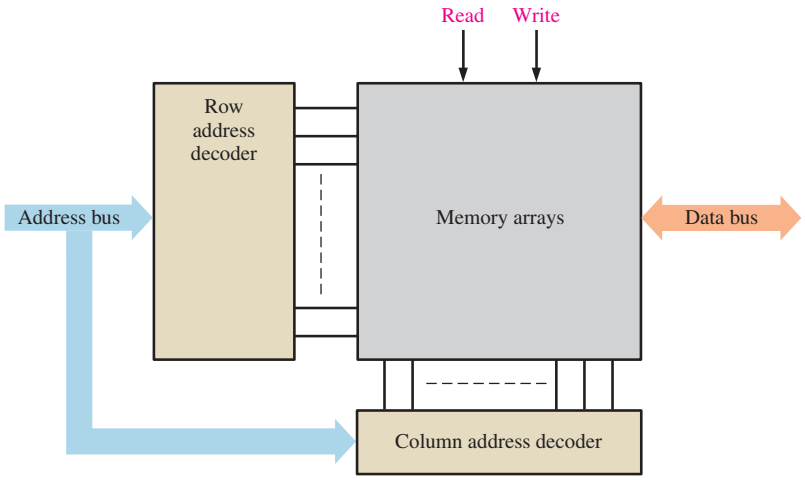
Data units go into the memory during a write operation and come out of the memory during a read operation on a set of lines called the *data bus*. As indicated in Figure 11–5, the data bus is bidirectional, which means that data can go in either direction (into the memory or out of the memory). In this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

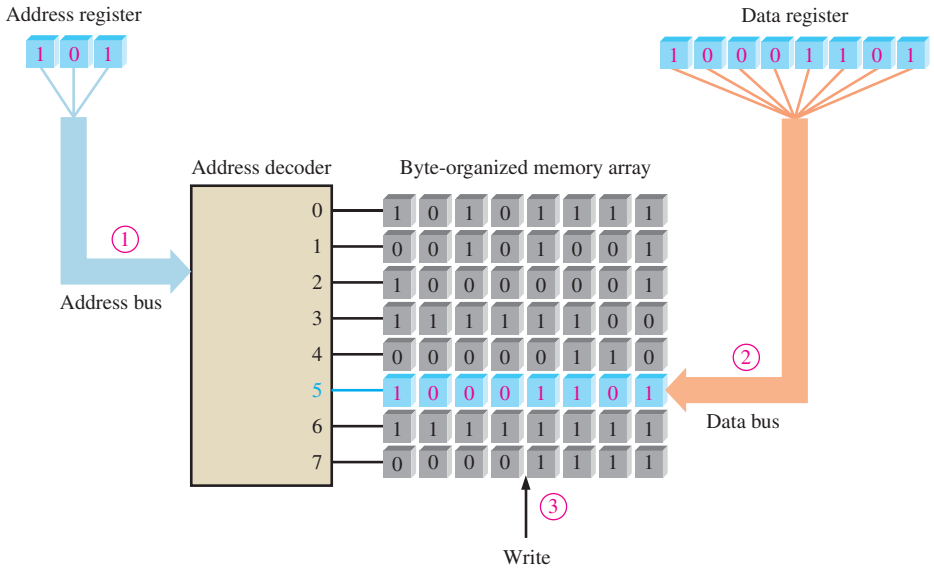


(a) Single-array memory



(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.

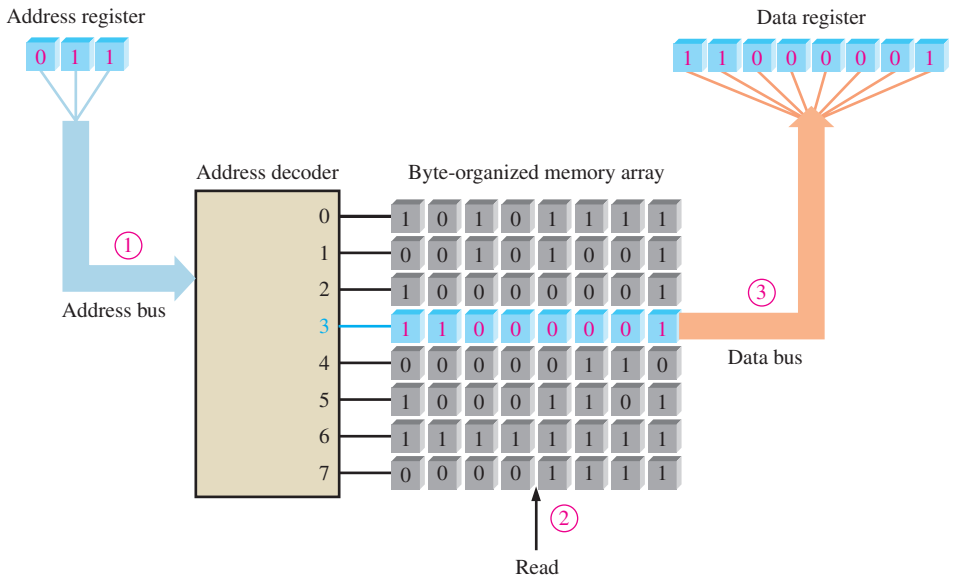


1. Address code 101 is placed on the address bus and address 5 is selected.
2. Data byte is placed on the data bus.
3. Write command causes the data byte to be stored in address 5, replacing previous data.

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.

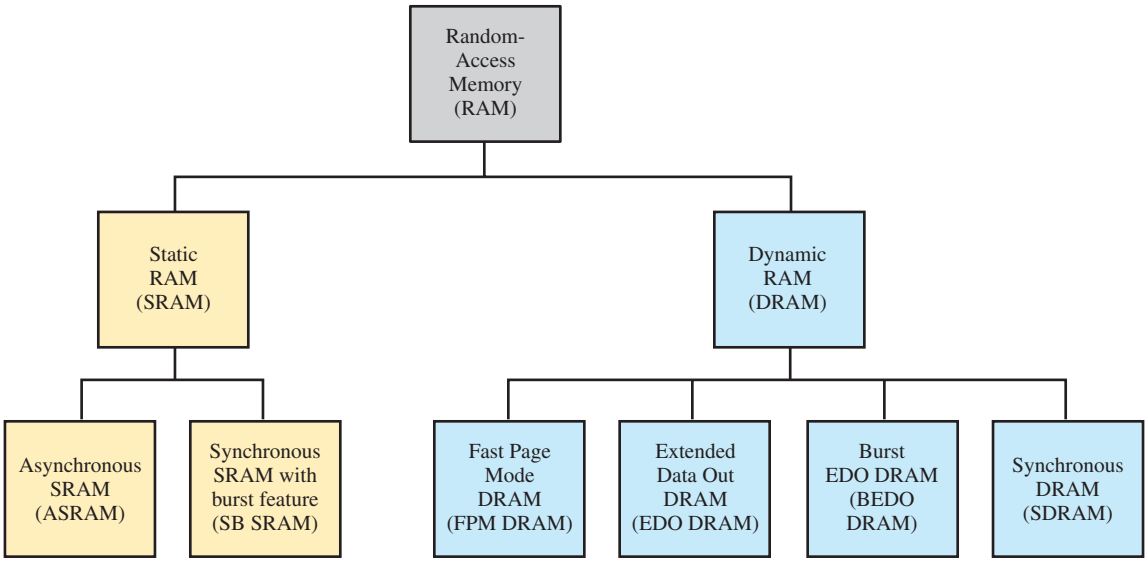


FIGURE 11–8 The RAM family.

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.

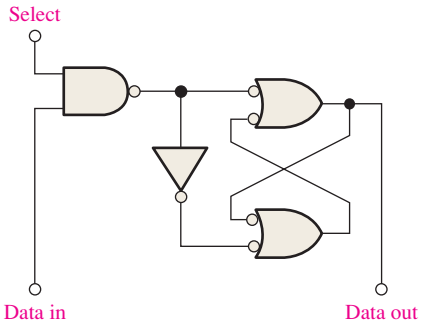


FIGURE 11–9 A typical SRAM latch memory cell.

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.

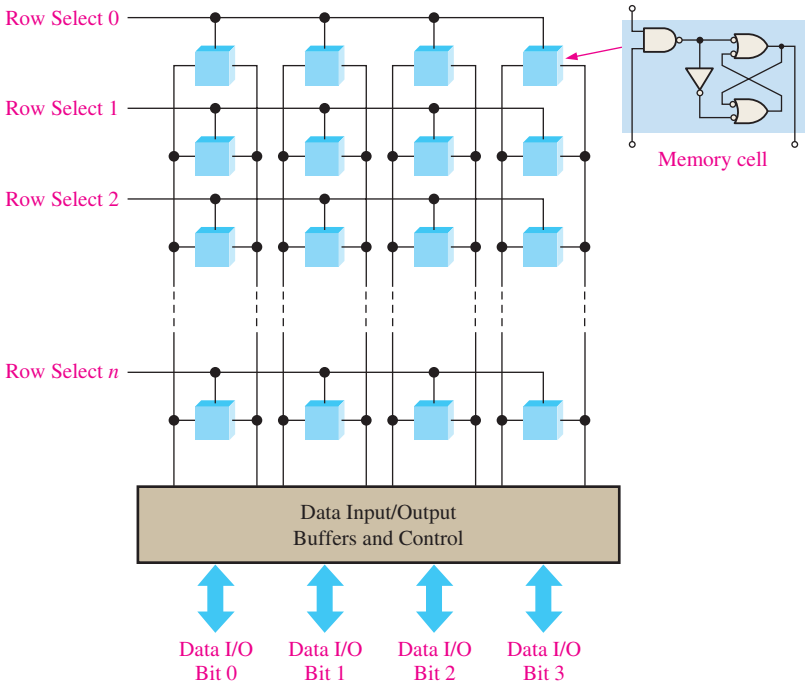


FIGURE 11–10 Basic SRAM array.

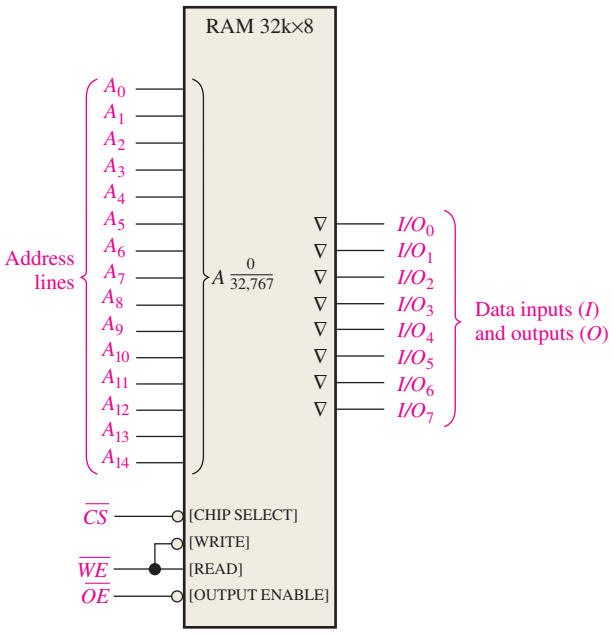


FIGURE 11–11 Logic diagram for an asynchronous 32k × 8 SRAM.

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

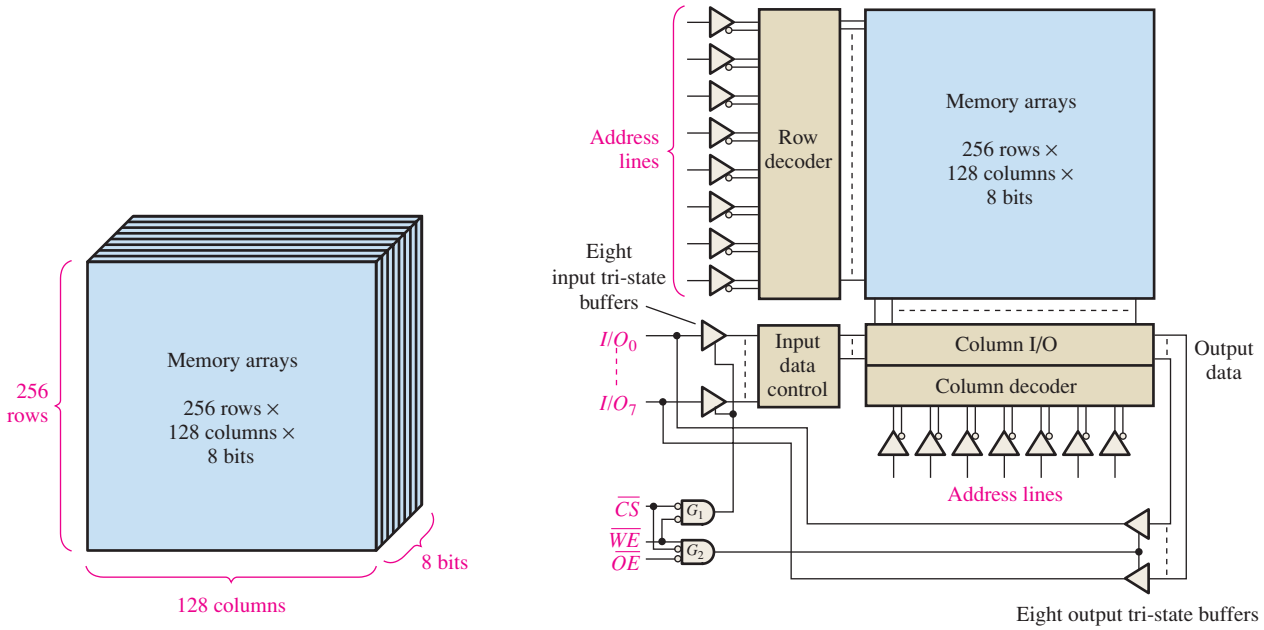
SRAM chips can be organized in single bits, nibbles (4 bits), bytes (8 bits), or multiple bytes (words with 16, 24, 32 bits, etc.).

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

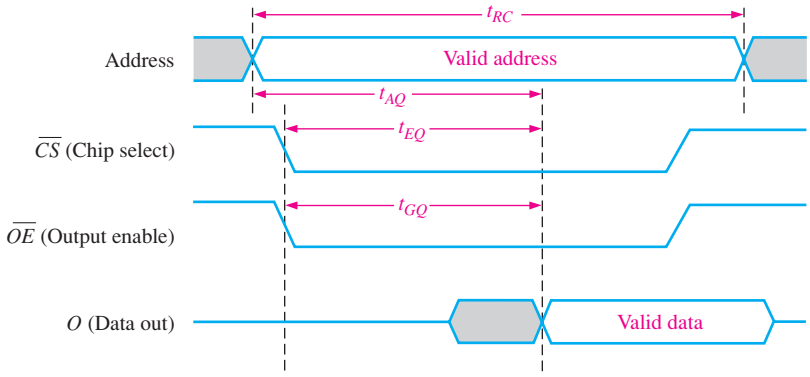
In the WRITE mode, $\overline{WE}$ is LOW and $\overline{OE}$ is HIGH. The input tri-state buffers are enabled by gate G_1, and the output tri-state buffers are disabled by gate G_2. Therefore, the eight input data bits on the data lines are routed through the input data control and the column I/O to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

Read and Write Cycles

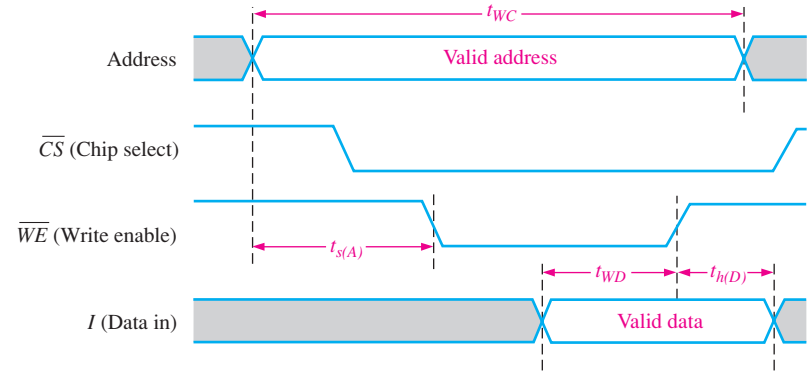
Figure 11–13 shows typical timing diagrams for a memory read cycle and a write cycle. For the read cycle shown in part (a), a valid address code is applied to the address lines for a specified time interval called the *read cycle time,* t_{RC}. Next, the chip select ($\overline{CS}$) and the output enable ($\overline{OE}$) 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.