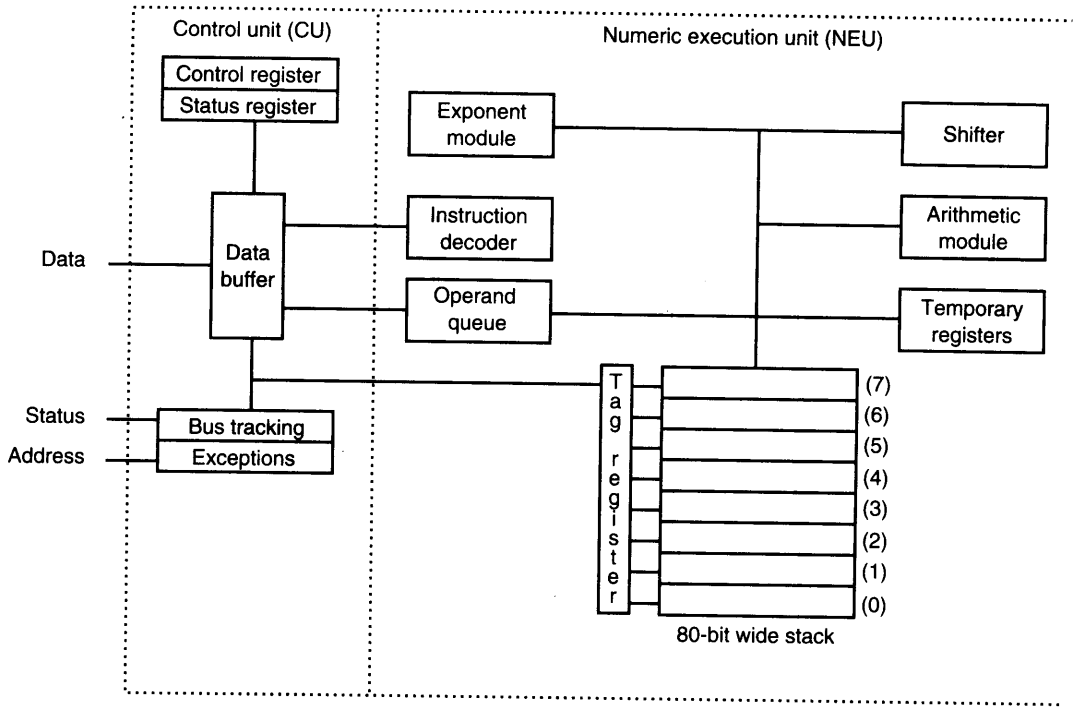

FIGURE 8–22 The internal structure of the 80X87 arithmetic coprocessor.

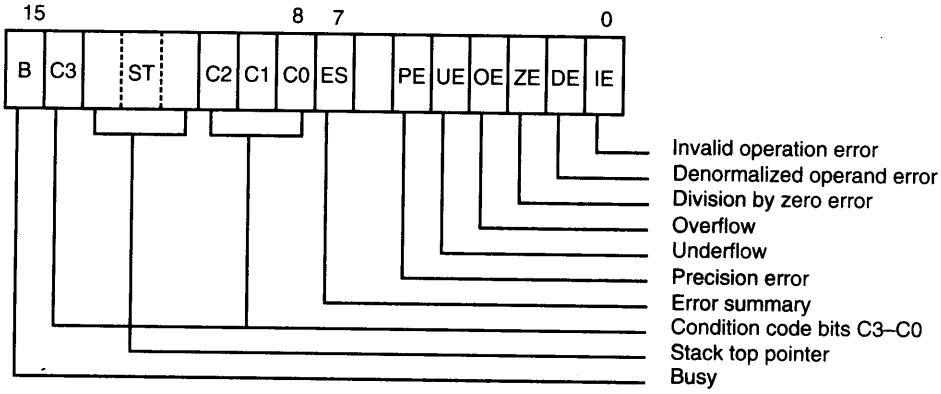

FIGURE 8–23 The 80X87 arithmetic coprocessor status register.

B The **busy bit** indicates that the coprocessor is busy executing a task. Busy can be tested by examining the status register or by using the FWAIT instruction. Newer coprocessors automatically synchronize with the microprocessor, so the busy flag need not be tested before performing additional coprocessor tasks.

C3–C0 The **condition code bits** indicate conditions about the coprocessor (see Table 8–8 for a complete listing of each combination of these bits and their functions). Note that these

TABLE 8–8 The 80X87 status register condition code bits.

Instruction	*C3*	*C2*	*C1*	*C0*	*Indication*
FTST, FCOM	0	0	X	0	ST > Operand
	0	0	X	1	ST < Operand
	1	0	X	1	ST = Operand
	1	1	X	1	ST is not comparable
FPREM	Q1	0	Q0	Q2	Rightmost 3 bits of quotient
	?	1	?	?	Incomplete
FXAM	0	0	0	0	+ unnormal
	0	0	0	1	+ NAN
	0	0	1	0	– unnormal
	0	0	1	1	– NAN
	0	1	0	0	+ normal
	0	1	0	1	+∞
	0	1	1	0	– normal
	0	1	1	1	–∞
	1	0	0	0	+ 0
	1	0	0	1	Empty
	1	0	1	0	– 0
	1	0	1	1	Empty
	1	1	0	0	+ denormal
	1	1	0	1	Empty
	1	1	1	0	– denormal
	1	1	1	1	Empty

Notes: Unnormal = leading bits of the significand are zero; denormal = exponent at its most negative value; normal = standard floating-point form; NAN (not-a-number) = an exponent of all ones and a significand not equal to zero, and the operand for FTST is zero.

bits have different meanings for different instructions, as indicated in the table. The top of the stack is denoted as ST in this table.

TOP The **top-of-stack (ST)** bit indicates the current register addressed as the top-of-the-stack (ST). This is normally register 0.

ES The **error summary** bit is set if any unmasked error bit (PE, UE, OE, ZE, DE, or IE) is set. In the 8087 coprocessor, the error summary also causes a coprocessor interrupt.

PE The **precision error** indicates that the result or operands exceed the selected precision.

UE An **underflow error** indicates a non-zero result that is too small to represent with the current precision selected by the control word.

OE An **overflow error** indicates a result that is too large to be represented. If this error is masked, the coprocessor generates infinity for an overflow error.

ZE A **zero error** indicates the divisor was zero while the dividend is a non-infinity or non-zero number.

DE A **denormalized error** indicates that at least one of the operands is denormalized.

IE An **invalid error** indicates a stack overflow or underflow, indeterminate form ($0 \div 0$, $+\infty$, $-\infty$, etc.), or the use of a NAN as an operand. This flag indicates errors such as those produced by taking the square root of a negative number, etc.

Control Register. The control register is pictured in Figure 8–24. The control register selects precision, rounding control, and infinity control. It also masks and unmasks the exception bits that correspond to the rightmost six bits of the status register. The FLDCW instruction is used to load a value into the control register. Following is a description of each bit or grouping of bits found in the control register:

IC	**Infinity control** selects either affine or projective infinity. Affine allows positive and negative infinity, while projective assumes infinity is unsigned.
RC	**Rounding control** determines the type of rounding, as defined in Figure 8–24.
PC	The **precision control** sets the precision of the result, as defined in Figure 8–24.
Exception Masks	Determine whether the error indicated by the exception affects the error bit in the status register. If a logic 1 is placed in one of the exception control bits, the corresponding status register bit is masked off.

Tag Register. The **tag register** indicates the contents of each location in the coprocessor stack. Figure 8–25 illustrates the tag register and the status indicated by each tag. The tag indicates whether a register is valid; zero; invalid or infinity; or empty. The only way that a program can view the tag register is by storing the coprocessor environment using the FSTENV, FSAVE, or FRSTOR instructions. Each of these instructions stores the tag register along with other coprocessor data..

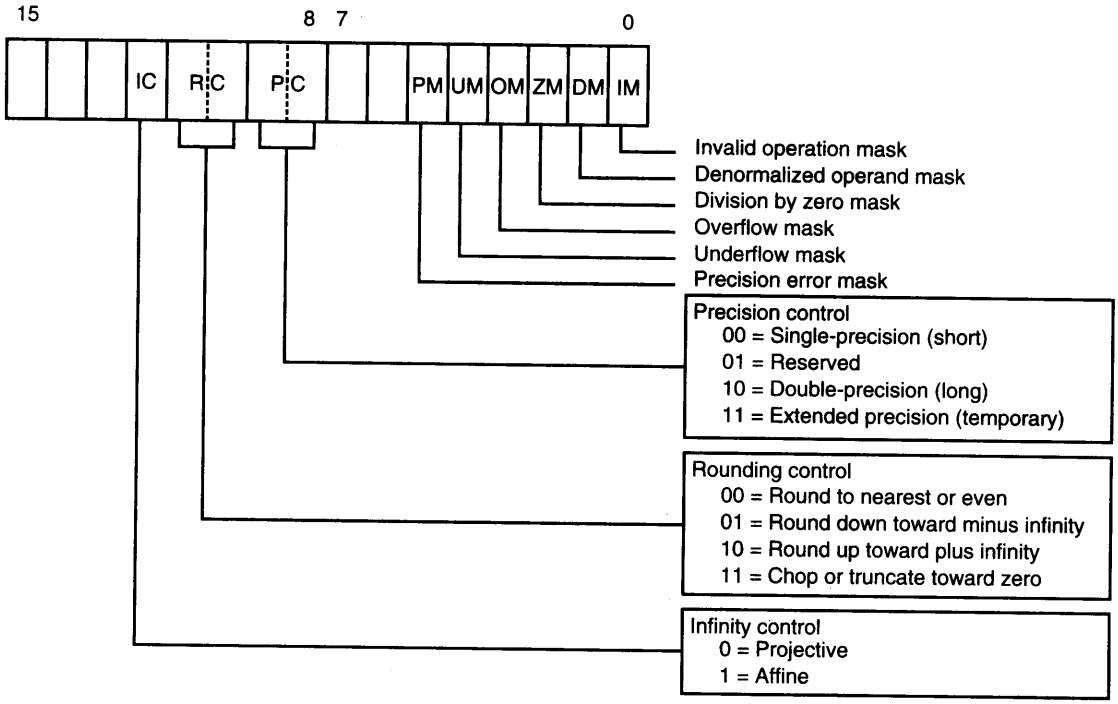

FIGURE 8–24 The 80X87 arithmetic coprocessor control register.

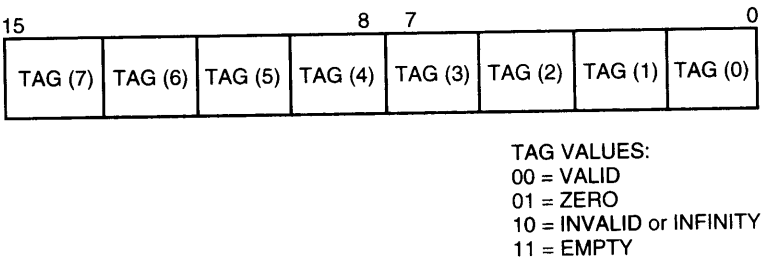

FIGURE 8–25 The 80X87 arithmetic coprocessor tag register.

Transcendental Operations

The transcendental instructions include FPTAN (partial tangent), FPATAN (partial arctangent), FSIN (sine), FCOS (cosine), FSINCOS (sine and cosine), F2XM1 ($2^x - 1$), FYL2X ($Y \log_2 X$), and FYL2XP1 ($Y \log_2 (X + 1)$). A list of these operations follows with a description of each transcendental operation:

FPTAN Finds the partial tangent of $Y/X = \tan \theta$. The value of θ is at the top of the stack. It must be between 0 and n/4 radians for the 8087. The result is a ratio found as ST = X and ST(1) = Y. If the value is outside of the allowable range, an invalid error occurs, as indicated by the status register IE bit. Also note that ST(7) must be empty for this instruction to function properly.

FPATAN Finds the partial arctangent as θ = ARCTAN X/Y. The value of X is at the top of the stack and Y is at ST(1). The values of X and Y must be as follows: $0 \leq Y < X < \infty$. The instruction pops the stack and leaves θ at the top of the stack.

F2XM1 Finds the function $2^x - 1$. The value of X is taken from the top of the stack and the result is returned to the top of the stack. To obtain 2^x, add one to the result at the top of the stack. The value of X must be in the range of −1 and +1. The F2XM1 instruction is used to derive the functions listed in Table 8-9. Note that the constants $\log_2 10$ and $\log_2 e$ are built-in as standard values for the coprocessor.

TABLE 8–9 Exponential functions.

Function	*Equation*
10^Y	$2^Y \times \log_2 10$
ε^Y	$2^Y \times \log_2 \varepsilon$
X^Y	$2^Y \times \log_2 X$

FSIN/FCOS Finds the sine or cosine of the argument located in ST expressed in radians (360° = 2π radians), with the result found in ST. The values of ST must be less than 2^{63}.

FSINCOS Finds the sine and cosine of ST, expressed in radians, and leaves the results as ST = sine and ST(1) = cosine.

FYL2X Finds $Y \log_2 X$. The value X is taken from the stack top, and Y is taken from ST(1). The result is found at the top of the stack after a pop. The value of X must range between 0 and ∞, and the value of Y must be between −∞ and +∞. A logarithm with any positive base (b) is found by the equation $LOG_b X = (LOG_2 b)^{-1} \times LOG_2 X$

FYL2XP1 Finds $Y \log_2 (X + 1)$. The value of X is taken from the stack top and Y is taken from ST(1). The result is found at the top of the stack after a pop. The value of X must range between 0 and $1 - \sqrt{1/2}$ and the value of Y must be between −∞ and +∞.

8–8 SUMMARY

1. The main differences between the 8086 and 8088 are (a) an eight-bit data bus on the 8088 and a 16-bit data bus on the 8086, (b) an SS0 pin on the 8088 in place of BHE/S7 on the 8086, and (c) an IO/M pin on the 8088 instead of an M/IO on the 8086.
2. Both the 8086 and 8088 require a single +5.0 V power supply with a tolerance of ±10 %.
3. The 8086/8088 microprocessors are TTL-compatible if the noise immunity Figure is de-rated to 350 mV from the customary 400 mV.
4. The 8086/8088 microprocessors can drive one 74XX, five 74LSXX, one 74SXX, ten 74ALSXX, and ten 74HCXX unit loads.
5. The 8284A clock generator provides the system clock (CLK), READY synchronization, and RESET synchronization.
6. The standard 5 MHz 8086/8088 operating frequency is obtained by attaching a 15 MHz crystal to the 8284A clock generator. The PCLK output contains a TTL-compatible signal at one-half the CLK frequency.
7. Whenever the 8086/8088 microprocessors are reset, they begin executing software at memory location FFFF0H (FFFF:0000) with the interrupt request pin disabled.
8. Because the 8086/8088 buses are multiplexed and most memory and I/O devices aren't, the system must be demultiplexed before interfacing with memory or I/O. Demultiplexing is accomplished by an eight-bit latch whose clock pulse is obtained from the ALE signal.
9. In a large system, the buses must be buffered because the 8086/8088 microprocessors are capable of driving only ten unit loads, and large systems often have many more.
10. Bus timing is very important to the remaining chapters in the text. A bus cycle that consists of four clocking periods acts as the basic system timing. Each bus cycle is able to read or write data between the microprocessor and the memory or I/O system.
11. A bus cycle is broken into four states, or T periods: T1 is used by the microprocessor to send the address to the memory or I/O and the ALE signal to the demultiplexers; T2 is used to send data to memory for a write and to test the READY pin and activate control signals RD or WR; T3 allows the memory time to access data and allows data to be transferred between the microprocessor and the memory or I/O; and T4 is where data are written.
12. The 8086/8088 microprocessors allow the memory and I/O 460 ns to access data when they are operated with a 5 MHz clock.
13. Wait states (T_W) stretch the bus cycle by one or more clocking periods to allow the memory and I/O additional access time. Wait states are inserted by controlling the READY input to the 8086/8088. READY is sampled at the end of T2 and during T_W.
14. Minimum mode operation is similar to that of the Intel 8085A microprocessor, while maximum mode operation is new and specifically designed for the operation of the 8087 arithmetic coprocessor.
15. The 8288 bus controller must be used in the maximum mode to provide the control bus signals to the memory and I/O. This is because the maximum mode operation of the 8086/8088 removes some of the system's control signal lines in favor of control signals for the coprocessors. The 8288 reconstructs these removed control signals.
16. The arithmetic coprocessor functions in parallel with the microprocessor. This means that the microprocessor and coprocessor can execute their respective instructions simultaneously.
17. The data types manipulated by the coprocessor include signed-integer, floating-point, and binary-coded decimal (BCD).
18. There are three forms of integers used with the coprocessor: word (16 bits), short (32 bits), and long (64 bits). Each integer contains a signed number in true magnitude for positive numbers and two's complement form for negative numbers.
19. The coprocessor supports three types of floating-point numbers: single-precision (32 bits), double-precision (64 bits), and extended-precision (80 bits). A floating-point number has three parts: the sign, biased exponent,

and significand. In the coprocessor, the exponent is biased with a constant and the integer bit of the normalized number is not stored in the significand, except in the extended-precision form.

20. The coprocessor contains a status register that indicates busy, what conditions follow a compare or test, the location of the top of the stack, and the state of the error bits. The FSTSW AX instruction, followed by SAHF, is often used with conditional jump instructions to test for some coprocessor conditions.
21. The control register of the coprocessor contains control bits that select infinity, rounding, precision, and error masks.
22. The following directives are often used with the coprocessor for storing data: DW (define word), DD (define doubleword), DQ (define quadword), and DT (define 10 bytes).
23. The coprocessor uses a stack to transfer data between itself and the memory system. Generally, data are loaded to the top of the stack or removed from the top of the stack for storage.
24. All internal coprocessor data are always in the 80-bit extended-precision form. The only time that data are in any other form is when they are stored or loaded from the memory.
25. The coprocessor addressing modes include the classic stack mode, register, register with a pop, and memory. Stack addressing is implied and the data at ST become the source, ST(1) the destination, and the result is found in ST after a pop.
26. The coprocessor's arithmetic operations include addition, subtraction, multiplication, division, and square root calculation.
27. There are transcendental functions in the coprocessor's instruction set. These functions find the partial tangent or arctangent, $2^x - 1$, $Y \log_2 X$, and $Y \log_2 (X + 1)$. .

8–9 QUESTIONS AND PROBLEMS

1. List the differences between the 8086 and the 8088 microprocessors.
2. Is the 8086/8088 TTL-compatible? Explain your answer.
3. What is the fan-out from the 8086/8088 to the following devices:
   (a) 74XXX TTL
   (b) 74ALSXXX TTL
   (c) 74HCXXX CMOS
4. What information appears on the address/data bus of the 8088 while ALE is active?
5. What are the purposes of status bits S3 and S4?
6. What condition does a logic 0 on the 8086/8088 RD pin indicate?
7. Explain the operation of the TEST pin and the WAIT instruction.
8. Describe the signal that is applied to the CLK input pin of the 8086/8088 microprocessors.
9. What mode of operation is selected when MN/MX is grounded?
10. What does the WR strobe signal from the 8086/8088 indicate about the operation of the 8086/8088?
11. When does ALE float to its high-impedance state?
12. When DT/R is a logic 1, what condition does it indicate about the operation of the 8086/8088?
13. What happens when the HOLD input to the 8086/8088 is placed at its logic 1 level?
14. What three minimum mode 8086/8088 pins are decoded to discover whether the processor is halted?
15. Explain the operation of the LOCK pin.
16. What conditions do the QS1 and QS0 pins indicate about the 8086/8088?
17. What three housekeeping chores are provided by the 8284A clock generator?
18. By what factor does the 8284A clock generator divide the crystal oscillator's output frequency?
19. If the F/C pin is placed at a logic 1 level, the crystal oscillator is disabled. Where is the timing input signal attached to the 8284A under this condition?
20. The PCLK output of the 8284A is _____ MHz if the crystal oscillator is operating at 14 MHz.

21. The RES input to the 8284A is placed at a logic _____ level in order to reset the 8086/8088.
22. Which bus connections on the 8086 microprocessor are typically demultiplexed?
23. Which bus connections on the 8088 microprocessor are typically demultiplexed?
24. Which TTL-integrated circuit is often used to demultiplex the buses on the 8086/8088?
25. What is the purpose of the demultiplexed BHE signal on the 8086 microprocessor?
26. Why are buffers often required in an 8086/8088-based system?
27. What 8086/8088 signal is used to select the direction of the data flows through the 74LS245 bi-directional bus buffer?
28. A bus cycle is equal to clocking ___ periods.
29. If the CLK input to the 8086/8088 is 4 MHz, how long is one bus cycle?
30. What two 8086/8088 operations occur during a bus cycle?
31. How many MIPS is the 8086/8088 capable of obtaining when operated with a 10 MHz clock?
32. Briefly describe the purpose of each T state listed:
    (a) T1
    (b) T2
    (c) T3
    (d) T4
33. How much time is allowed for memory access when the 8086/8088 is operated with a 5 MHz clock?
34. How wide is DEN if the 8088 is operated with a 5 MHz clock?
35. If the READY pin is grounded, it will introduce _____ states into the bus cycle of the 8086/8088.
36. What does the ASYNC input to the 8284A accomplish?
37. What logic levels must be applied to AEN1 and RDY1 to obtain a logic 1 at the READY pin? (Assume that AEN2 is at a logic 1 level.)
38. Contrast minimum and maximum mode 8086/8088 operation.
39. What main function is provided by the 8288 bus controller when used with 8086/8088 maximum mode operation?
40. Explain what the microprocessor does when a coprocessor instruction executes.
41. What is the purpose of the C3–C0 bits in the status register?
42. What operation is accomplished with the FSTSW AX instruction?
43. What is the purpose of the IE bit in the status register?
44. How can SAHF and a conditional jump instruction be used to determine whether the top of the stack (ST) is equal to register ST(2)?
45. How is the rounding mode selected in the 80X87?
46. What coprocessor instruction uses the microprocessor's AX register?
47. What I/O ports are reserved for coprocessor use with the 80287?
48. How are data stored inside the coprocessor?
49. What is a NAN?
50. Whenever the coprocessor is reset, the top of the stack register is register number _____.
51. What does the term *chop* mean in the rounding control bits of the control register?
52. What is the difference between affine and projective infinity control?

CHAPTER 9

Memory Interface

INTRODUCTION

Whether simple or complex, every microprocessor-based system has a memory system. The Intel family of microprocessors is no different from any other in this respect.

Almost all systems contain two main types of memory: read-only memory (ROM) and random access memory (RAM) or read/write memory. Read-only memory contains system software and permanent system data, while RAM contains temporary data and application software. This chapter explains how to interface both memory types to the Intel family of microprocessors. We demonstrate memory interface to an 8-, 16-, and 32-bit data bus by using various memory address sizes. This allows virtually any microprocessor to be inter-faced to any memory system.

CHAPTER OBJECTIVES

Upon completion of this chapter, you will be able to:

1. Decode the memory address and use the outputs of the decoder to select various memory components.
2. Use programmable logic devices (PLDs) to decode memory addresses.
3. Explain how to interface both RAM and ROM to a microprocessor.
4. Interface memory to an 8-, 16-, 32-, and 64-bit data bus.
5. Interface dynamic RAM to the microprocessor.

9–1 MEMORY DEVICES

Before attempting to interface memory to the microprocessor, it is essential to completely understand the operation of memory components. In this section, we explain the functions of the four common types of memory: **read-only memory** (ROM), **flash memory** (EEPROM), **static random access memory** (SRAM), and **dynamic random access memory** (DRAM).

Memory Pin Connections

Pin connections common to all memory devices are the address inputs, data outputs or input/ outputs, some type of selection input, and at least one control input used to select a read or write operation. See Figure 9–1 for ROM and RAM generic-memory devices.

Address Connections. All memory devices have address inputs that select a memory location within the memory device. Address inputs are almost always labeled from A0, the least-significant address input, to A_n, where subscript n can be any value but is always labeled as one less than the total number of address pins. For example, a memory device with 10 address pins has its address pins labeled from A0 to A9. The number of address pins found on a memory device is determined by the number of memory locations found within it.

Today, the more common memory devices have between 1K (1024) to 64M (67,108,864) memory locations, with 256M memory location devices on the horizon. A 1K memory device has 10 address pins (A0–A9); therefore, 10 address inputs are required to select any of its 1024 memory locations. It takes a 10-bit binary number (1024 different combinations) to select any single location on a 1024-location device. If a memory device has 11 address connections (A0–A10), it has 2048 (2K) internal memory locations. The number of memory locations can thus be extrapolated from the number of address pins. For example, a 4K memory device has 12 address connections, an 8K device has 13, and so forth. A device that contains 1M locations requires a 20-bit address (A0–A19).

A 400H represents a 1K-byte section of the memory system. If a memory device is decoded to begin at memory address 10000H and it is a 1K device, its last location is at address 103FFH—one location less than 400H. Another important hexadecimal number to remember is a 1000H, because 1000H is 4K. A memory device that contains a starting address of 14000H that is 4K bytes long, ends at location 14FFFH—one location less than 1000H. A third number is 64K, or 10000H. A memory that starts at location 30000H and ends at location 3FFFFH is a 64K byte memory. Finally, because 1M of memory is common, a 1M memory contains 100000H memory locations.

Data Connections. All memory devices have a set of data outputs or input/outputs. The device illustrated in Figure 9–1 has a common set of input/output (I/O) connections. Today, many memory devices have bi-directional common I/O pins.

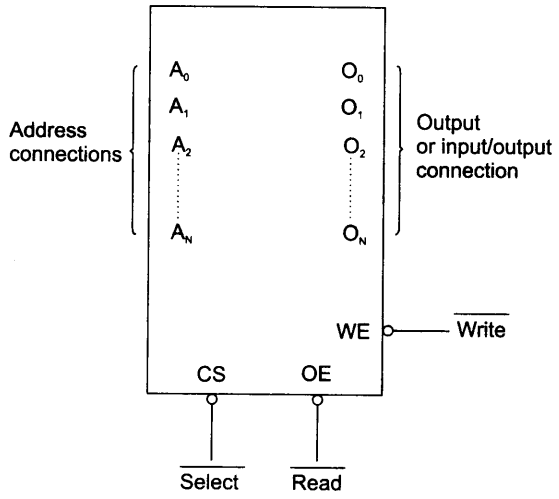

FIGURE 9–1 A pseudomemory component illustrating the address, data, and control connections.

The data connections are the points at which data are entered for storage or extracted for reading. Data pins on memory devices are labeled D0 through D7 for an 8-bit-wide memory device. In this sample memory device, there are eight I/O connections, which means that the memory device stores eight bits of data in each of its memory locations. An 8-bit-wide memory device is often called a byte-wide memory. Although most devices are currently eight bits wide, some devices are 16 bits, four bits, or just one bit wide.

Catalog listings of memory devices often refer to memory locations times bits per location. For example, a memory device with 1K memory locations and eight bits in each location is often listed as a 1K × 8 by the manufacturer. A 16K × 1 is a memory device containing 16K 1-bit memory locations. Memory devices are often classified according to total bit capacity. For example, a 1K × 8-bit memory device is sometimes listed as an 8K memory device, or a 64K × 4 memory is listed as a 256K device. These variations occur from one manufacturer to another.

Selection Connections. Each memory device has an input—sometimes more than one—that selects or enables the memory device. This type of input is most often called a **chip select** (CS), **chip enable** (CE), or simply **select** (S) input. RAM memory generally has at least one CS or S input, and ROM has at least one CE. If the CE, CS, or S input is active (a logic 0, in this case, because of the over-bar), the memory device performs a read or write operation; if it is inactive (a logic 1, in this case), the memory device cannot do a read or a write because it is turned off or disabled. If more than one CS connection is present, all must be activated to read or write data.

Control Connections. All memory devices have some form of control input or inputs. A ROM usually has only one control input, while a RAM often has one or two control inputs.
The control input most often found on a ROM is the **output enable** (OE) or **gate** (G) connection, which allows data to flow out of the output data pins of the ROM. If OE and the selection input (CS) are both active, the output is enabled; if OE is inactive, the output is disabled at its high-impedance state. The OE connection enables and disables a set of three-state buffers located within the memory device and must be active to read data.

A RAM memory device has either one or two control inputs. If there is only one control input, it is often called R/W. This pin selects a read operation or a write operation only if the device is selected by the selection input (CS). If the RAM has two control inputs, they are usually labeled WE (or W), and OE (or G). Here, WE **(write enable)** must be active to perform a memory write, and OE must be active to perform a memory read operation. When these two controls (WE and OE) are present, they must never both be active at the same time. If both control inputs are inactive (logic 1s), data are neither written nor read, and the data connections are at their high-impedance state.

ROM Memory

The **read-only memory** (ROM) permanently stores programs and data that are resident to the system and must not change when power supply is disconnected. The ROM is permanently programmed so that data are always present, even when power is disconnected. This type of memory is often called **nonvolatile memory.**

Today, the ROM is available in many forms. A device we call a ROM is purchased in mass quantities from a manufacturer and programmed during its fabrication at the factory. The EPROM **(erasable programmable read-only memory),** a type of ROM, is more commonly used when software must be changed often or when too few are in demand to make the ROM economical. For a ROM to be practical, we usually must purchase at least 10,000 devices to recoup the factory programming charge. An EPROM is programmed in the field on a device called an *EPROM programmer.* The EPROM is also erasable if exposed to high-intensity ultraviolet light for about 20 minutes or less, depending on the type of EPROM.

PROM memory devices are also available, although they are not as common today. The PROM **(programmable read-only memory)** is also programmed in the field by burning open tiny NI-chrome or silicon oxide fuses; but once it is programmed, it cannot be erased.

Still another, newer type of **read-mostly memory** (RMM) is called the **flash memory.** The flash memory[1] is also often called an EEPROM **(electrically erasable programmable ROM),** EAROM **(electrically alterable**

[1]Flash memory is a registered trademark of Intel Corporation.