- ☐ List the steps performed by the 8085 microprocessor, and identify the contents of buses when an instruction is being executed.
- ☐ Analyze a memory interfacing circuit, and specify the memory addresses of a given memory device.
- ☐ Recognize partial decoding and identify foldback (mirror) memory space.

4.1 THE 8085 MPU

The term **microprocessing unit** (MPU) is similar to the term **central processing unit** (CPU) used in traditional computers. We define the MPU as a device or a group of devices (as a unit) that can communicate with peripherals, provide timing signals, direct data flow, and perform computing tasks as specified by the instructions in memory. The unit will have the necessary lines for the address bus, the data bus, and the control signals, and would require only a power supply and a crystal (or equivalent frequency-determining components) to be completely functional.

Using this description, the 8085 microprocessor can almost qualify as an MPU, but with the following two limitations.

1. The low-order address bus of the 8085 microprocessor is **multiplexed** (time-shared) with the data bus. The buses need to be demultiplexed.
2. Appropriate control signals need to be generated to interface memory and I/O with the 8085. (Intel has some specialized memory and I/O devices that do not require such control signals.)

This section shows how to demultiplex the bus and generate the control signals after describing the 8085 microprocessor and illustrates the bus timings.

4.1.1 The 8085 Microprocessor

The 8085A (commonly known as the 8085) is an 8-bit general-purpose microprocessor capable of addressing 64K of memory. The device has forty pins, requires a +5 V single power supply, and can operate with a 3-MHz single-phase clock. The 8085A-2 version can operate at the maximum frequency of 5 MHz. The 8085 is an enhanced version of its predecessor, the 8080A; its instruction set is upward-compatible with that of the 8080A, meaning that the 8085 instruction set includes all the 8080A instructions plus some additional ones.

Figure 4.1 shows the logic pinout of the 8085 microprocessor. All the signals can be classified into six groups: (1) address bus, (2) data bus, (3) control and status signals, (4) power supply and frequency signals, (5) externally initiated signals, and (6) serial I/O ports.

ADDRESS BUS

The 8085 has 16 signal lines (pins) that are used as the address bus; however, these lines are split into two segments: A_{15}–A_8 and AD_7–AD_0. The eight signal lines, A_{15}–A_8, are

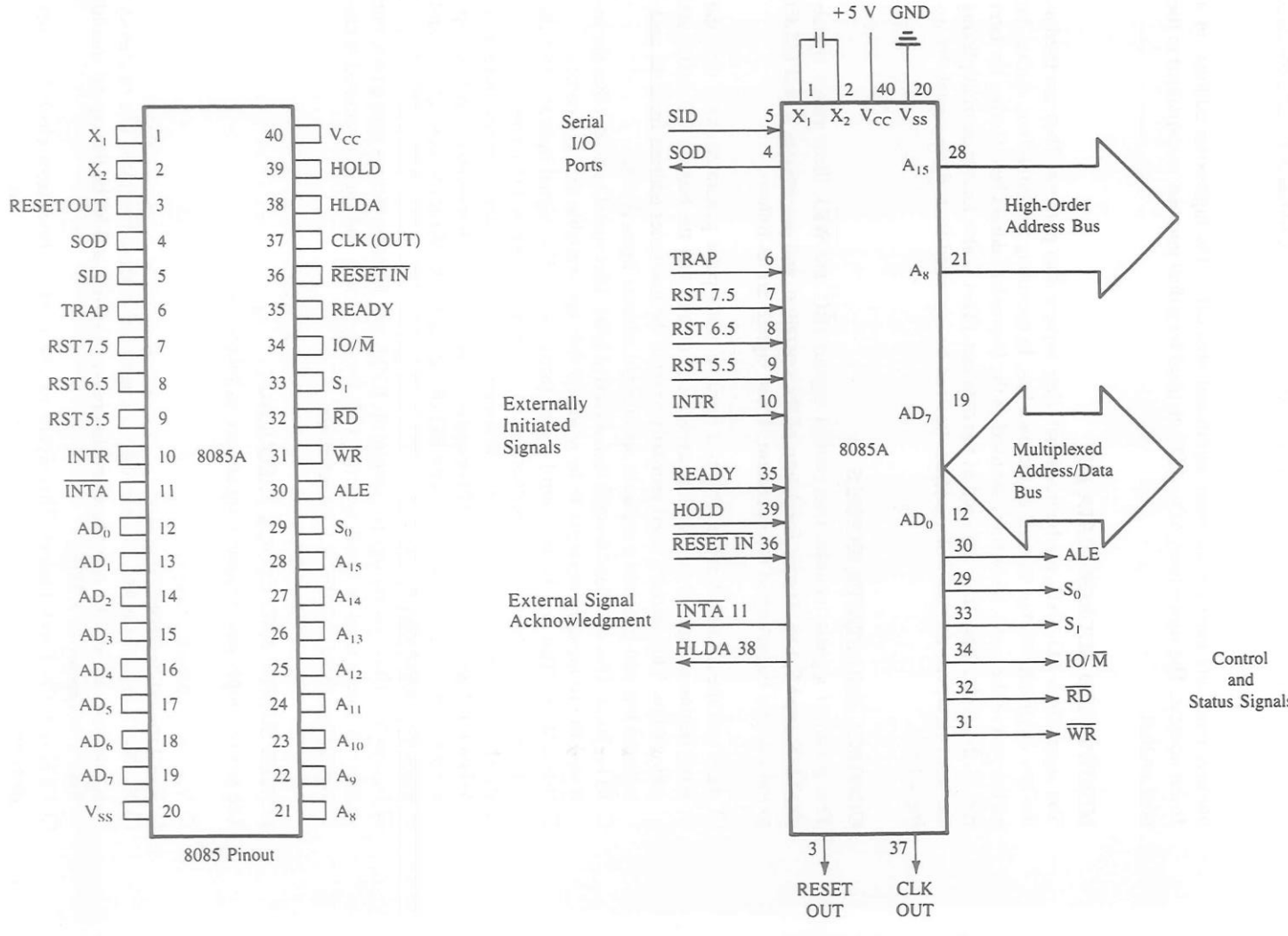


FIGURE 4.1
The 8085 Microprocessor Pinout and Signals

NOTE: The 8085A is commonly known as the 8085.
SOURCE (Pinout): Intel Corporation, *Embedded Microprocessors* (Santa Clara, Calif.: Author, 1994), pp. 1-11.

unidirectional and used for the most significant bits, called the high-order address, of a 16-bit address. The signal lines AD_7–AD_0 are used for a dual purpose, as explained in the next section.

MULTIPLEXED ADDRESS/DATA BUS

The signal lines AD_7–AD_0 are bidirectional: they serve a dual purpose. They are used as the low-order address bus as well as the data bus. In executing an instruction, during the earlier part of the cycle, these lines are used as the low-order address bus. During the later part of the cycle, these lines are used as the data bus. (This is also known as multiplexing the bus.) However, the low-order address bus can be separated from these signals by using a latch.

CONTROL AND STATUS SIGNALS

This group of signals includes two control signals ($\overline{RD}$ and $\overline{WR}$), three status signals ($IO/\overline{M}$, S_1, and S_0) to identify the nature of the operation, and one special signal (ALE) to indicate the beginning of the operation. These signals are as follows:

- ALE—Address Latch Enable: This is a positive going pulse generated every time the 8085 begins an operation (machine cycle); it indicates that the bits on AD_7–AD_0 are address bits. This signal is used primarily to latch the low-order address from the multiplexed bus and generate a separate set of eight address lines, A_7–A_0.
- $\overline{RD}$—Read: This is a Read control signal (active low). This signal indicates that the selected I/O or memory device is to be read and data are available on the data bus.
- $\overline{WR}$—Write: This is a Write control signal (active low). This signal indicates that the data on the data bus are to be written into a selected memory or I/O location.
- $IO/\overline{M}$: This is a status signal used to differentiate between I/O and memory operations. When it is high, it indicates an I/O operation; when it is low, it indicates a memory operation. This signal is combined with $\overline{RD}$ (Read) and $\overline{WR}$ (Write) to generate I/O and memory control signals.
- S_1 and S_0: These status signals, similar to $IO/\overline{M}$, can identify various operations, but they are rarely used in small systems. (All the operations and their associated status signals are listed in Table 4.1 for reference.)

POWER SUPPLY AND CLOCK FREQUENCY

The power supply and frequency signals are as follows:

- V_{CC}: +5 V power supply.
- V_{SS}: Ground Reference.
- X_1, X_2: A crystal (or RC, LC network) is connected at these two pins. The frequency is internally divided by two; therefore, to operate a system at 3 MHz, the crystal should have a frequency of 6 MHz.
- CLK (OUT)—Clock Output: This signal can be used as the system clock for other devices.

TABLE 4.1
8085 Machine Cycle Status and Control Signals

Machine Cycle	Status			Control Signals
	IO/$\overline{M}$	S_1	S_0	
Opcode Fetch	0	1	1	$\overline{RD} = 0$
Memory Read	0	1	0	$\overline{RD} = 0$
Memory Write	0	0	1	$\overline{WR} = 0$
I/O Read	1	1	0	$\overline{RD} = 0$
I/O Write	1	0	1	$\overline{WR} = 0$
Interrupt Acknowledge	1	1	1	$\overline{INTA} = 0$
Halt	Z	0	0	$\overline{RD}$, $\overline{WR}$ = Z and $\overline{INTA} = 1$
Hold	Z	X	X	
Reset	Z	X	X	

NOTE: Z = Tri-state (high impedance)
X = Unspecified

EXTERNALLY INITIATED SIGNALS, INCLUDING INTERRUPTS

The 8085 has five interrupt signals (see Table 4.2) that can be used to interrupt a program execution. One of the signals, INTR (Interrupt Request), is identical to the 8080A microprocessor interrupt signal (INT); the others are enhancements to the 8080A. The microprocessor acknowledges an interrupt request by the $\overline{INTA}$ (Interrupt Acknowledge) signal. (The interrupt process is discussed in Chapter 12.)

In addition to the interrupts, three pins—RESET, HOLD, and READY—accept the externally initiated signals as inputs. To respond to the HOLD request, the 8085 has one

TABLE 4.2
8085 Interrupts and Externally Initiated Signals

☐ INTR (Input)	Interrupt Request: This is used as a general-purpose interrupt; it is similar to the INT signal of the 8080A.
☐ $\overline{INTA}$ (Output)	Interrupt Acknowledge: This is used to acknowledge an interrupt.
☐ RST 7.5 (Inputs) RST 6.5 RST 5.5	Restart Interrupts: These are vectored interrupts that transfer the program control to specific memory locations. They have higher priorities than the INTR interrupt. Among these three, the priority order is 7.5, 6.5, and 5.5.
☐ TRAP (Input)	This is a nonmaskable interrupt and has the highest priority.
☐ HOLD (Input)	This signal indicates that a peripheral such as a DMA (Direct Memory Access) controller is requesting the use of the address and data buses.
☐ HLDA (Output)	Hold Acknowledge: This signal acknowledges the HOLD request.
☐ READY (Input)	This signal is used to delay the microprocessor Read or Write cycles until a slow-responding peripheral is ready to send or accept data. When this signal goes low, the microprocessor waits for an integral number of clock cycles until it goes high.

signal called HLDA (Hold Acknowledge). The functions of these signals were previously discussed in Section 3.1.3. The RESET is again described below, and others are listed in Table 4.2 for reference.

- $\overline{\text{RESET IN}}$: When the signal on this pin goes low, the program counter is set to zero, the buses are tri-stated, and the MPU is reset.
- RESET OUT: This signal indicates that the MPU is being reset. The signal can be used to reset other devices.

SERIAL I/O PORTS

The 8085 has two signals to implement the serial transmission: SID (Serial Input Data) and SOD (Serial Output Data). In serial transmission, data bits are sent over a single line, one bit at a time, such as the transmission over telephone lines. This will be discussed in Chapter 16 on serial I/O.

In this chapter, we will focus on the first three groups of signals; others will be discussed in later chapters.

4.1.2 Microprocessor Communication and Bus Timings

To understand the functions of various signals of the 8085, we should examine the process of communication (reading from and writing into memory) between the microprocessor and memory and the timings of these signals in relation to the system clock. The first step in the communication process is reading from memory or fetching an instruction. This can be easily understood using an analogy of how a package is picked up from your house by a shipping company such as Federal Express. The steps are as follows:

1. A courier gets the address from the office; he or she drives the pickup van, finds the street, and looks for your house number.
2. The courier rings the bell.
3. Somebody in the house opens the door and gives the package to the courier, and the courier returns to the office with the package.
4. The internal office staff disposes the package according to the instructions given by the customer.

Now let us examine the steps in the following example of how the microprocessor fetches or gets a machine code from memory.

Example 4.1 Refer to Example 3.5 in the last chapter (Section 3.2.6): Illustrate the steps and the timing of data flow when the instruction code 0100 1111 (4FH—MOV C,A), stored in location 2005H, is being fetched.

Solution To fetch the byte (4FH), the MPU needs to identify the memory location 2005H and enable the data flow from memory. This is called the Fetch cycle. The data flow is shown in Figure 4.2, and the timings are explained below.

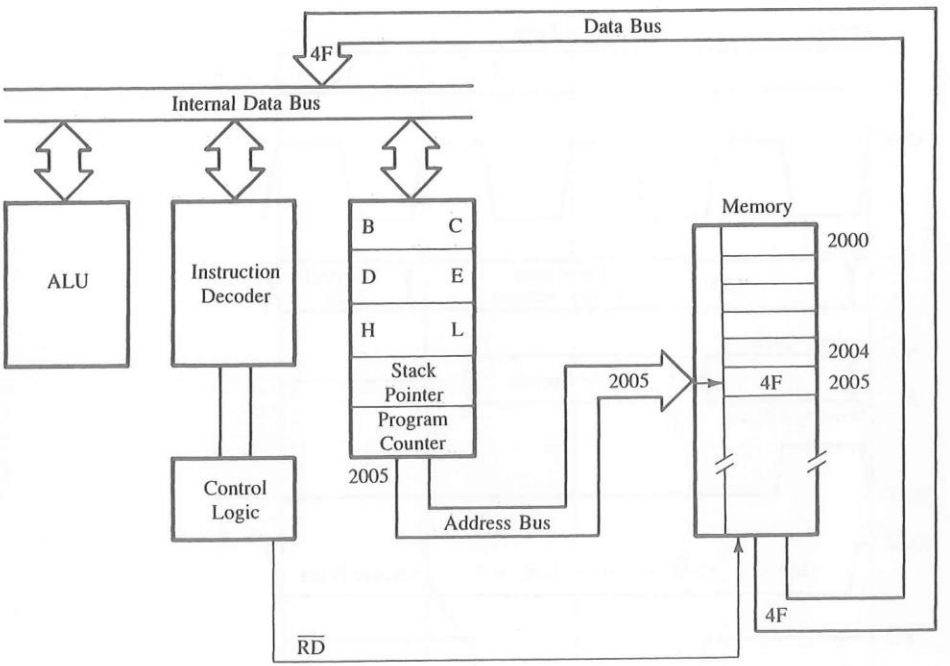


FIGURE 4.2
Data Flow from Memory to the MPU

Figure 4.3 shows the timing of how a data byte is transferred from memory to the MPU; it shows five different groups of signals in relation to the system clock. The address bus and data bus are shown as two parallel lines. This is a commonly used practice to represent logic levels of groups of lines; some lines are high and others are low. The crossover of the lines indicates that a new byte (information) is placed on the bus, and a dashed straight line indicates the high impedance state. To fetch the byte, the MPU performs the following steps:

Step 1: The microprocessor places the 16-bit memory address from the program counter (PC) on the address bus (Figure 4.2). In our analogy, this is the equivalent of our courier getting on the road to find the address.

Figure 4.3 shows that at T_1 the high-order memory address 20H is placed on the address lines A_{15}–A_8, the low-order memory address 05H is placed on the bus AD_7–AD_0, and the ALE signal goes high. Similarly, the status signal IO/$\overline{M}$ goes low, indicating that this is a memory-related operation. (For the sake of clarity, the other two status signals, S_1 and S_0, are not shown in Figure 4.3; they will be discussed in the next section.)

Step 2: The control unit sends the control signal $\overline{RD}$ to enable the memory chip (Figure 4.2). This is similar to ringing the doorbell in our analogy of a package pickup.

The control signal $\overline{RD}$ is sent out during the clock period T_2, thus enabling the memory chip (Figure 4.3). The $\overline{RD}$ signal is active during two clock periods.

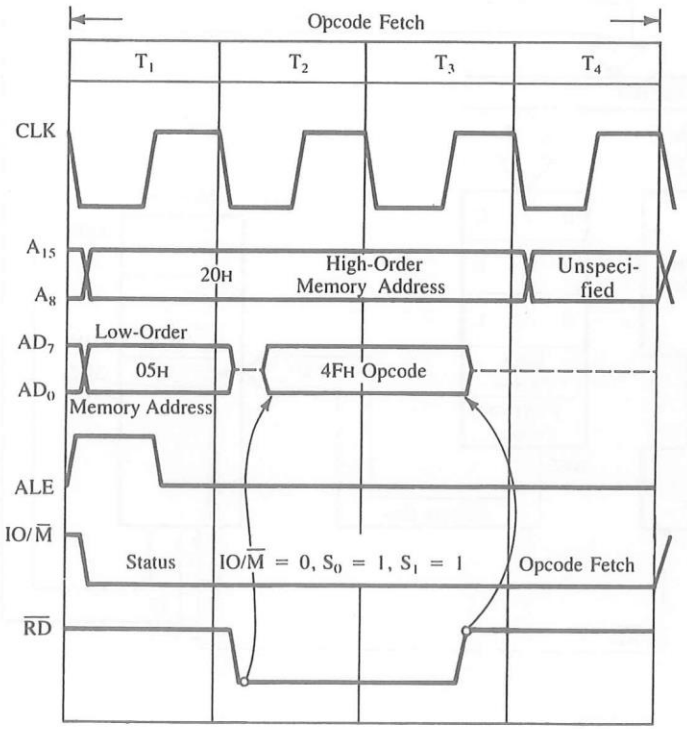


FIGURE 4.3
Timing: Transfer of Byte from Memory to MPU

Step 3: The byte from the memory location is placed on the data bus.

When the memory is enabled, the instruction byte (4FH) is placed on the bus AD_7–AD_0 and transferred to the microprocessor. The $\overline{RD}$ signal causes 4FH to be placed on bus AD_7–AD_0 (shown by the arrow), and when $\overline{RD}$ goes high, it causes the bus to go into high impedance.

Step 4: The byte is placed in the instruction decoder of the microprocessor, and the task is carried out according to the instruction.

The machine code or the byte (4FH) is decoded by the instruction decoder, and the contents of the accumulator are copied into register C. This task is performed during the period T_4 in Figure 4.3.

The above four steps are similar to the steps listed in our analogy of the package pickup.

4.1.3 Demultiplexing the Bus AD_7–AD_0

The need for demultiplexing the bus AD_7–AD_0 becomes easier to understand after examining Figure 4.3. This figure shows that the address on the high-order bus (20H) remains on the bus for three clock periods. However, the low-order address (05H) is lost after the

first clock period. This address needs to be latched and used for identifying the memory address. If the bus AD_7–AD_0 is used to identify the memory location (2005H), the address will change to 204FH after the first clock period.

Figure 4.4 shows a schematic that uses a latch and the ALE signal to demultiplex the bus. The bus AD_7–AD_0 is connected as the input to the latch 74LS373. The ALE signal is connected to the Enable (G) pin of the latch, and the Output control ($\overline{OC}$) signal of the latch is grounded.

Figure 4.3 shows that the ALE goes high during T_1. When the ALE is high, the latch is transparent; this means that the output changes according to input data. During T_1, the output of the latch is 05H. When the ALE goes low, the data byte 05H is latched until the next ALE, and the output of the latch represents the low-order address bus A_7–A_0 after the latching operation.

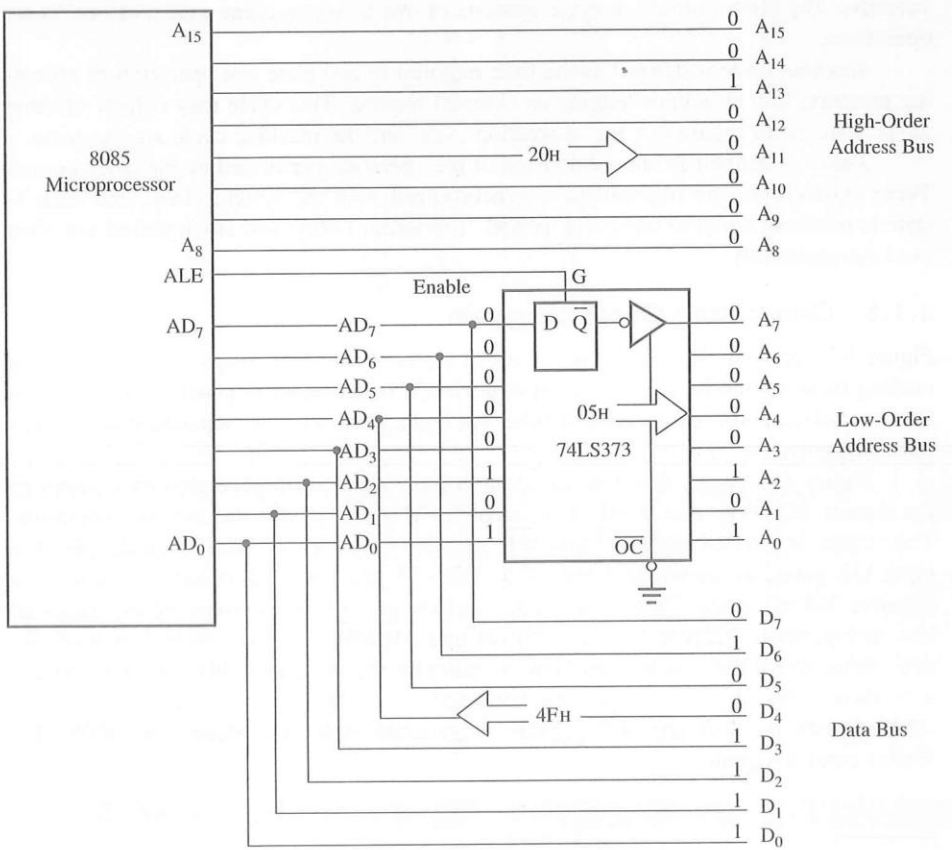


FIGURE 4.4
Schematic of Latching Low-Order Address Bus

Intel has circumvented the problem of demultiplexing the low-order bus by designing special devices such as the 8155 (256 bytes of R/W memory + I/Os), which is compatible with the 8085 multiplexed bus. These devices internally demultiplex the bus using the ALE signal (see Figures 4.18 and 4.19).

After carefully examining Figure 4.3, we can make the following observations:

1. The machine code 4FH (0100 1000) is a one-byte instruction that copies the contents of the accumulator into register C.
2. The 8085 microprocessor requires one external operation—fetching a machine code* from memory location 2005H.
3. The entire operation—fetching, decoding, and executing—requires four clock periods.

Now we can define three terms—instruction cycle, machine cycle, and T-state—and use these terms later for examining timings of various 8085 operations (Section 4.2).

Instruction cycle is defined as the time required to complete the execution of an instruction. The 8085 instruction cycle consists of one to six machine cycles or one to six operations.

Machine cycle is defined as the time required to complete one operation of accessing memory, I/O, or acknowledging an external request. This cycle may consist of three to six T-states. In Figure 4.3, the instruction cycle and the machine cycle are the same.

T-state is defined as one subdivision of the operation performed in one clock period. These subdivisions are internal states synchronized with the system clock, and each T-state is precisely equal to one clock period. The terms T-state and clock period are often used synonymously.

4.1.4 Generating Control Signals

Figure 4.3 shows the $\overline{RD}$ (Read) as a control signal. Because this signal is used both for reading memory and for reading an input device, it is necessary to generate two different Read signals: one for memory and another for input. Similarly, two separate Write signals must be generated.

Figure 4.5 shows that four different control signals are generated by combining the signals $\overline{RD}$, $\overline{WR}$, and IO/$\overline{M}$. The signal IO/$\overline{M}$ goes low for the memory operation. This signal is ANDed with $\overline{RD}$ and $\overline{WR}$ signals by using the 74LS32 quadruple two-input OR gates, as shown in Figure 4.5. The OR gates are functionally connected as negative NAND gates. When both input signals go low, the outputs of the gates go low and generate $\overline{MEMR}$ (Memory Read) and $\overline{MEMW}$ (Memory Write) control signals. When the IO/$\overline{M}$ signal goes high, it indicates the peripheral I/O operation. Figure 4.5 shows that this signal is complemented using the Hex inverter 74LS04 and ANDed with the $\overline{RD}$ and $\overline{WR}$ signals to generate $\overline{IOR}$ (I/O Read) and $\overline{IOW}$ (I/O Write) control signals.

---

*This code is an operation code (opcode) that instructs the microprocessor to perform the specified task. The term *opcode* was explained in Chapter 2 (Section 2.3).

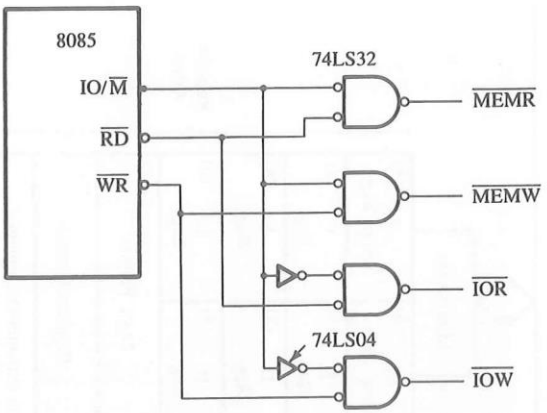


FIGURE 4.5
Schematic to Generate Read/Write Control Signals for Memory and I/O

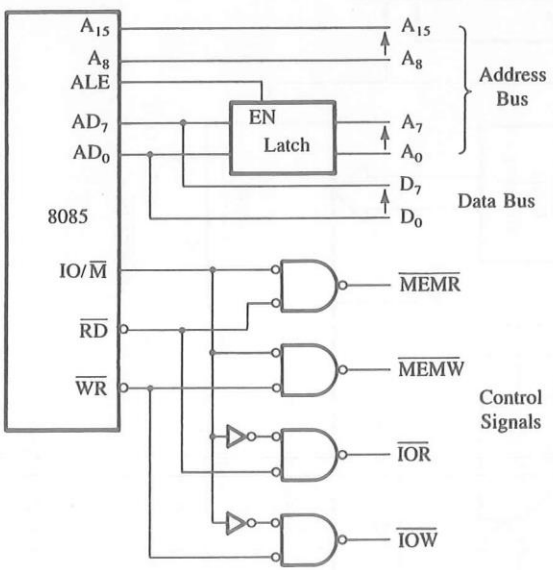


FIGURE 4.6
8085 Demultiplexed Address and Data Bus with Control Signals

To demultiplex the bus and to generate the necessary control signals, the 8085 microprocessor requires a latch and logic gates to build the MPU, as shown in Figure 4.6. This MPU can be interfaced with any memory or I/O.

4.1.5 A Detailed Look at the 8085 MPU and Its Architecture

Figure 4.7 shows the internal architecture of the 8085 beyond the programmable registers we discussed previously. It includes the ALU (Arithmetic/Logic Unit), Timing and

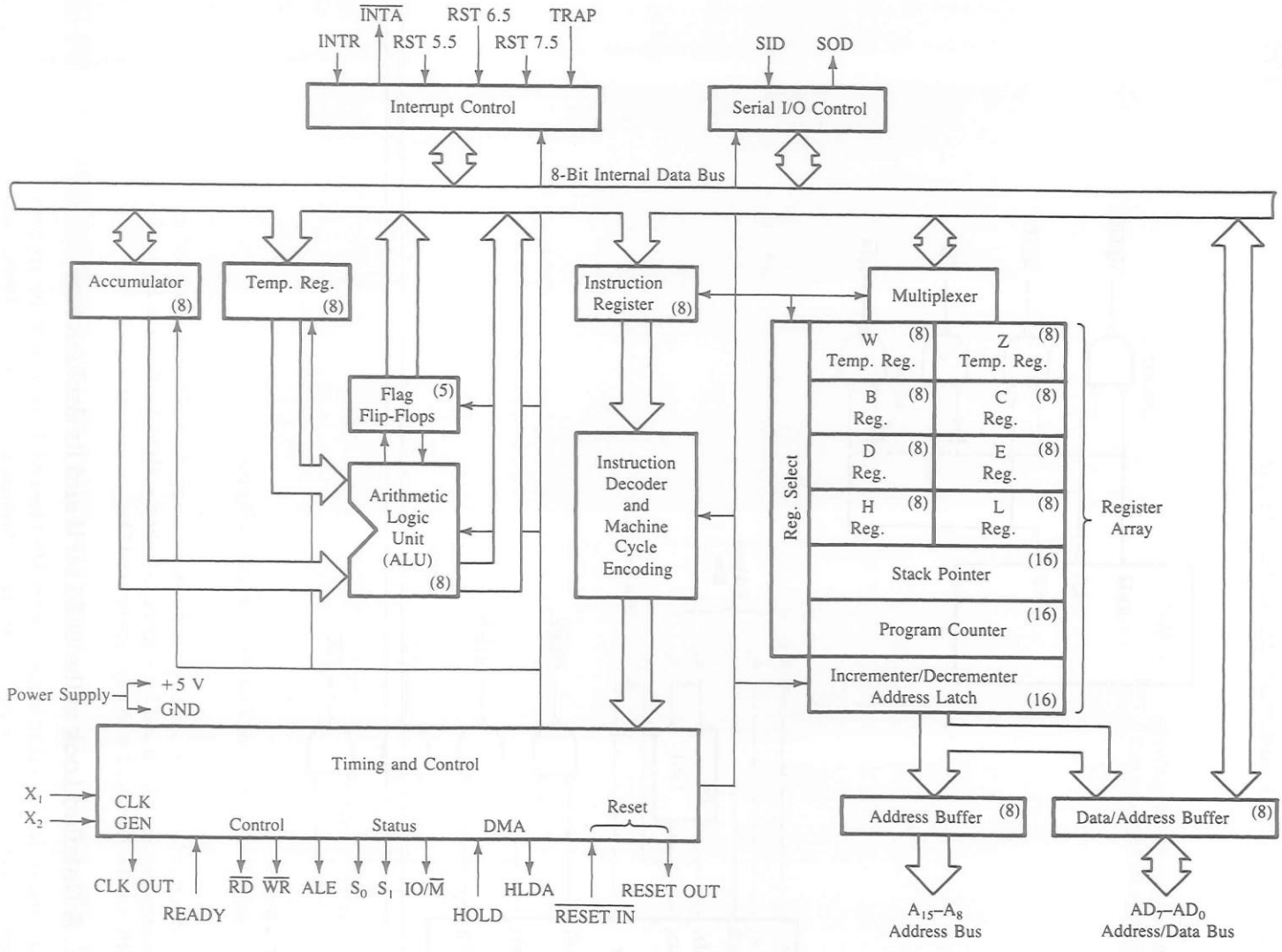


FIGURE 4.7
The 8085A Microprocessor: Functional Block Diagram

NOTE: The 8085A microprocessor is commonly known as the 8085.

SOURCE: Intel Corporation, *Embedded Microprocessors* (Santa Clara, Calif.: Author, 1994), pp. 1-11.

Control Unit, Instruction Register and Decoder, Register Array, Interrupt Control, and Serial I/O Control. We will discuss the first four units below; the last two will be discussed later in the book.

THE ALU

The arithmetic/logic unit performs the computing functions; it includes the accumulator, the temporary register, the arithmetic and logic circuits, and five flags. The temporary register is used to hold data during an arithmetic/logic operation. The result is stored in the accumulator, and the flags (flip-flops) are set or reset according to the result of the operation.

The flags are affected by the arithmetic and logic operations in the ALU. In most of these operations, the result is stored in the accumulator. Therefore, the flags generally reflect data conditions in the accumulator—with some exceptions. The descriptions and conditions of the flags are as follows:

- **S—Sign flag:** After the execution of an arithmetic or logic operation, if bit D_7 of the result (usually in the accumulator) is 1, the Sign flag is set. This flag is used with signed numbers. In a given byte, if D_7 is 1, the number will be viewed as a negative number; if it is 0, the number will be considered positive. In arithmetic operations with signed numbers, bit D_7 is reserved for indicating the sign, and the remaining seven bits are used to represent the magnitude of a number. However, this flag is irrelevant for the operations of unsigned numbers. Therefore, for unsigned numbers, even if bit D_7 of a result is 1 and the flag is set, it does not mean the result is negative. (See Appendix A2 for a discussion of signed numbers.)
- **Z—Zero flag:** The Zero flag is set if the ALU operation results in 0, and the flag is reset if the result is not 0. This flag is modified by the results in the accumulator as well as in the other registers.
- **AC—Auxiliary Carry flag:** In an arithmetic operation, when a carry is generated by digit D_3 and passed on to digit D_4, the AC flag is set. The flag is used only internally for BCD (binary-coded decimal) operations and is not available for the programmer to change the sequence of a program with a jump instruction.
- **P—Parity flag:** After an arithmetic or logical operation, if the result has an even number of 1s, the flag is set. If it has an odd number of 1s, the flag is reset. (For example, the data byte 0000 0011 has even parity even if the magnitude of the number is odd.)
- **CY—Carry flag:** If an arithmetic operation results in a carry, the Carry flag is set; otherwise it is reset. The Carry flag also serves as a borrow flag for subtraction.

The bit positions reserved for these flags in the flag register are as follows:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z		AC		P		CY

Among the five flags, the AC flag is used internally for BCD arithmetic; the instruction set does not include any conditional jump instructions based on the AC flag. Of the remaining four flags, the Z and CY flags are those most commonly used.

TIMING AND CONTROL UNIT

This unit synchronizes all the microprocessor operations with the clock and generates the control signals necessary for communication between the microprocessor and peripherals. The control signals are similar to a sync pulse in an oscilloscope. The $\overline{RD}$ and $\overline{WR}$ signals are sync pulses indicating the availability of data on the data bus.

INSTRUCTION REGISTER AND DECODER

The instruction register and the decoder are part of the ALU. When an instruction is fetched from memory, it is loaded in the instruction register. The decoder decodes the instruction and establishes the sequence of events to follow. The instruction register is not programmable and cannot be accessed through any instruction.

REGISTER ARRAY

The programmable registers were discussed in the last chapter. Two additional registers, called temporary registers W and Z, are included in the register array. These registers are used to hold 8-bit data during the execution of some instructions. However, because they are used internally, they are not available to the programmer.

4.1.6 Decoding and Executing an Instruction

Decoding and executing an instruction after it has been fetched can be illustrated with the example from Section 4.12.

Example 4.2 Assume that the accumulator contains data byte 82H, and the instruction MOV C,A (4FH) is fetched. List the steps in decoding and executing the instruction.

Solution This example is similar to the example in Section 4.12, except that the contents of the accumulator are specified. To decode and execute the instruction, the following steps are performed.

The microprocessor:

1. Places the contents of the data bus (4FH) in the instruction register (Figure 4.8) and decodes the instruction.
2. Transfers the contents of the accumulator (82H) to the temporary register in the ALU.
3. Transfers the contents of the temporary register to register C.

4.1.7 Review of Important Concepts

1. The 8085 microprocessor has a multiplexed bus AD_7–AD_0 used as the lower-order address bus and the data bus.
2. The bus AD_7–AD_0 can be demultiplexed by using a latch and the ALE signal.
3. The 8085 has a status signal IO/$\overline{M}$ and two control signals $\overline{RD}$ and $\overline{WR}$. By ANDing these signals, four control signals can be generated: $\overline{MEMR}$, $\overline{MEMW}$, $\overline{IOR}$, and $\overline{IOW}$.

The 8085 MPU module (Figure 4.9) includes devices such as the 8085 microprocessor, an octal latch, and logic gates, as shown previously in Figure 4.6. The octal latch demultiplexes the bus AD_7–AD_0 using the signal ALE, and the logic gates generate the necessary control signals. Figure 4.9 shows the demultiplexed address bus, the data bus, and the four active low control signals: $\overline{MEMR}$, $\overline{MEMW}$, $\overline{IOR}$, and $\overline{IOW}$. In addition, to increase the driving capacity of the buses, a unidirectional bus driver is used for the address bus and a bidirectional bus driver is used for the data bus. Now we can examine various operations the 8085 microprocessor performs in terms of machine cycles and T-states.

4.2.1 The 8085 Machine Cycles and Bus Timings

The 8085 microprocessor is designed to execute 74 different instruction types. Each instruction has two parts: operation code, known as opcode, and operand. The opcode is a command such as Add, and the operand is an object to be operated on, such as a byte or the contents of a register. Some instructions are 1-byte instructions and some are multibyte instructions. To execute an instruction, the 8085 needs to perform various operations such as Memory Read/Write and I/O Read/Write. However, there is no direct relationship between the number of bytes of an instruction and the number of operations the 8085 has to perform (this will be clarified later). In preceding sections, numerous 8085 signals and their functions were described. Now we need to examine these signals in conjunction with execution of individual instructions and their operations. This task may appear overwhelming at the beginning; fortunately, all instructions are divided into a few basic machine cycles and these machine cycles are divided into precise system clock periods.

Basically, the microprocessor external communication functions can be divided into three categories:

1. Memory Read and Write
2. I/O Read and Write
3. Request Acknowledge

These functions are further divided into various operations (machine cycles), as shown in Table 4.1. Each instruction consists of one or more of these machine cycles, and each machine cycle is divided into T-states.

In this section, we will focus on the first three operations listed in Table 4.1—Opcode Fetch, Memory Read, and Memory Write—and examine the signals on various buses in relation to the system clock. In the next section, we will use these timing diagrams to interface memory with the 8085 microprocessor. Similarly, we will discuss timings of other machine cycles in later chapters in the context of their applications. For example, I/O Read/Write machine cycles will be discussed in Chapter 5 on I/O interfacing, and Interrupt Acknowledge will be discussed in Chapter 12, "Interrupts."

4.2.2 Opcode Fetch Machine Cycle

The first operation in any instruction is Opcode Fetch. The microprocessor needs to get (fetch) this machine code from the memory register where it is stored before the microprocessor can begin to execute the instruction.

We discussed this operation in Example 4.1. Figure 4.2 shows how the 8085 fetches the machine code, using the address and the data buses and the control signal. Figure 4.3 shows the timing of the Opcode Fetch machine cycle in relation to the system's clock. In this operation, the processor reads a machine code (4FH) from memory. However, to differentiate an opcode from a data byte or an address, this machine cycle is identified as the Opcode Fetch cycle by the status signals ($IO/\overline{M} = 0$, $S_1 = 1$, $S_0 = 1$); the active low $IO/\overline{M}$ signal indicates that it is a memory operation, and S_1 and S_0 being high indicate that it is an Opcode Fetch cycle.

This Opcode Fetch cycle is called the M_1 cycle and has four T-states. The 8085 uses the first three states T_1–T_3 to fetch the code and T_4 to decode and execute the opcode. In the 8085 instruction set, some instructions have opcodes with six T-states. When we study the example of the Memory Read machine cycle, discussed in the next section, we may find that these two operations (Opcode Fetch and Memory/Read) are almost identical except that the Memory Read Cycle has three T-states.

4.2.3 Memory Read Machine Cycle

To illustrate the Memory Read machine cycle, we need to examine the execution of a 2-byte or a 3-byte instruction because in a 1-byte instruction the machine code is an opcode; therefore, the operation is always an Opcode Fetch. The execution of a 2-byte instruction is illustrated in the next example.

Example 4.3

Two machine codes—0011 1110 (3EH) and 0011 0010 (32H)—are stored in memory locations 2000H and 2001H, respectively, as shown below. The first machine code (3EH) represents the opcode to load a data byte in the accumulator, and the second code (32H) represents the data byte to be loaded in the accumulator. Illustrate the bus timings as these machine codes are executed. Calculate the time required to execute the Opcode Fetch and the Memory Read cycles and the entire instruction cycle if the clock frequency is 2 MHz.

Memory Location	Machine Code		Instruction
2000H	0 0 1 1 1 1 1 0	→ 3EH	MVI A,32H ;Load byte 32H
2001H	0 0 1 1 0 0 1 0	→ 32H	; in the accu-
			; mulator

Solution

This instruction consists of two bytes; the first is the opcode and the second is the data byte. The 8085 needs to read these bytes first from memory and thus requires at least two machine cycles. The first machine cycle is Opcode Fetch and the second machine cycle is Memory Read, as shown in Figure 4.10; this instruction requires seven T-states for these two machine cycles. The timings of the machine cycles are described in the following paragraphs.

1. The first machine cycle M_1 (Opcode Fetch) is identical in bus timings with the machine cycle illustrated in Example 4.1, except for the bus contents.

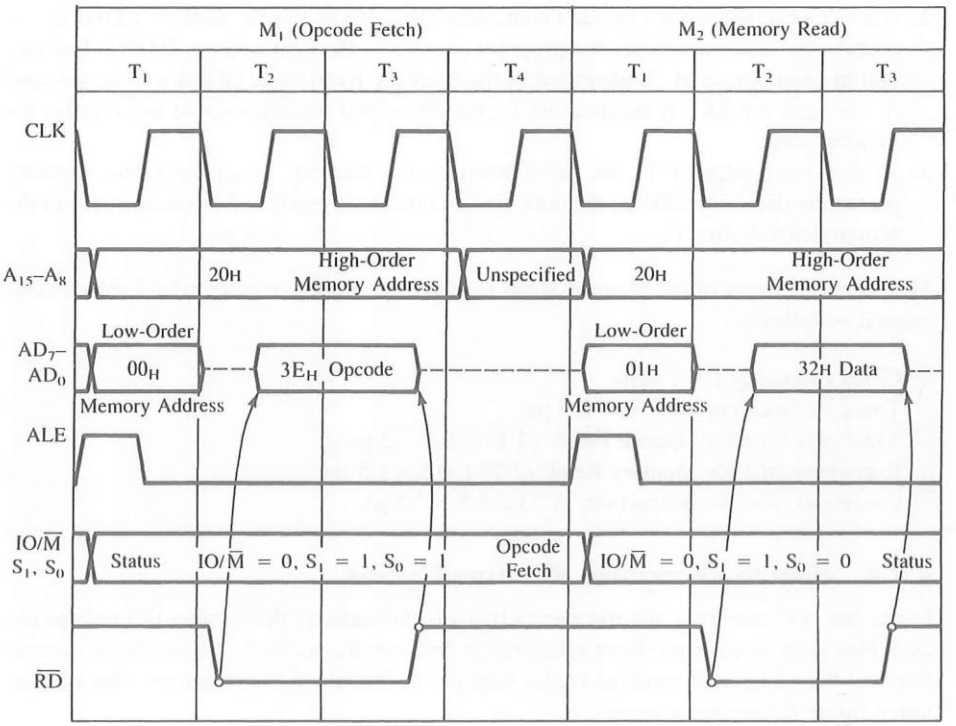


FIGURE 4.10
8085 Timing for Execution of the Instruction MVI A,32H

At T_1, the microprocessor identifies that it is an Opcode Fetch cycle by placing 011 on the status signals* ($IO/\overline{M} = 0$, $S_1 = 1$ and $S_0 = 1$). It places the memory address (2000H) from the program counter on the address bus, 20H on A_{15}–A_8, and 00H on AD_7–AD_0 and increments the program counter to 2001H to point to the next machine code. The ALE signal goes high during T_1, which is used to latch the low-order address 00H from the bus AD_7–AD_0. At T_2, the 8085 asserts the $\overline{RD}$ control signal, which enables the memory, and the memory places the byte 3EH from location 2000H on the data bus. Then the 8085 places the opcode in the instruction register and disables the $\overline{RD}$ signal. The fetch cycle is completed in state T_3. During T_4, the 8085 decodes the opcode and finds out that a second byte needs to be read. After the T_3 state, the contents of the bus A_{15}–A_8 are unknown, and the data bus AD_7–AD_0 goes into high impedance.

*The status signals S_1 and S_0 can be used to differentiate between various machine cycles. However, they are rarely needed; the necessary control signals can be generated by using $IO/\overline{M}$, $\overline{RD}$, and $\overline{WR}$.

2. After completion of the Opcode Fetch cycle, the 8085 places the address 2001H on the address bus and increments the program counter to the next address 2002H. The second machine cycle M_2 is identified as the Memory Read cycle ($IO/\overline{M} = 0$, $S_1 = 1$, and $S_0 = 0$) and the ALE is asserted. At T_2, the $\overline{RD}$ signal becomes active and enables the memory chip.
3. At the rising edge of T_2, the 8085 activates the data bus as an input bus, memory places the data byte 32H on the data bus, and the 8085 reads and stores the byte in the accumulator during T_3.

The execution times of the Memory Read machine cycle and the instruction cycle are calculated as follows:

- ☐ Clock frequency f = 2 MHz
- ☐ T-state = clock period (1/f) = 0.5 μs
- ☐ Execution time for Opcode Fetch: (4 T) × 0.5 = 2 μs
- ☐ Execution time for Memory Read: (3 T) × 0.5 = 1.5 μs
- ☐ Execution time for Instruction: (7 T) × 0.5 = 3.5 μs

4.2.4 How to Recognize Machine Cycles

In the last two examples, the number of bytes is the same as the number of machine cycles. However, there is no direct relationship between the number of bytes in an instruction and the number of machine cycles required to execute that instruction. This is illustrated in the following example.

Example 4.4

Explain the machine cycles of the following 3-byte instruction when it is executed.

Opcode	Operand	Bytes	Machine Cycles	T-States	Operation
STA	2065H	3	4	13	This instruction stores (writes) the contents of the accumulator in memory location 2065H

The machine codes are stored in memory locations 2010H, 2011H, and 2012H as follows: the 16-bit address of the operand must be entered in reverse order, the low-order byte first, followed by the high-order byte.

Memory Address	Machine Code		
2010	0011	0010 → 32H	Opcode
2011	0110	0101 → 65H	Low-order address
2012	0010	0000 → 20H	High-order address

Solution

This is a 3-byte instruction; however, it has four machine cycles with 13 T-states. The first operation in the execution of an instruction must be an Opcode Fetch. The 8085 requires three T-states for each subsequent operation; thus, nine T-states are required for the remaining machine cycles. Therefore, the Opcode Fetch in this instruction must be a four T-state machine cycle, the same as the one described in the previous example. The two machine cycles following the Opcode Fetch must be Memory Read machine cycles because the microprocessor must read all the machine codes (three bytes) before it can execute the instruction. Now let us examine what the instruction does. It stores (writes) the contents of the accumulator in memory location 2065H; therefore, the last machine cycle must be Memory Write. The execution steps are as follows:

1. In the first machine cycle, the 8085 places the address 2010H on the address bus and fetches the opcode 32H.
2. The second machine cycle is Memory Read. The processor places the address 2011H and gets the low-order byte 65H.
3. The third machine cycle is also Memory Read; the 8085 gets the high-order byte 20H from memory location 2012H.
4. The last machine cycle is Memory Write. The 8085 places the address 2065H on the address bus, identifies the operation as Memory Write ($IO/\overline{M} = 0$, $S_1 = 0$, and $S_0 = 1$). It places the contents of the accumulator on the data bus AD_7–AD_0 and asserts the $\overline{WR}$ signal. During the last T-state, the contents of the data bus are placed in memory location 2065H.

4.2.5 Review of Important Concepts

1. In each instruction cycle, the first operation is always Opcode Fetch. This cycle can be of four to six T-states duration.
2. The Memory Read cycle requires three T-states and is in many ways similar to the Opcode Fetch cycle. Both use the same control signal ($\overline{RD}$) and read contents from memory. However, the Opcode Fetch reads opcodes and the Memory Read reads 8-bit data or address; these two machine cycles are differentiated by the status signals.
3. When the status signal $IO/\overline{M}$ is active low, the 8085 indicates that it is a memory-related operation, and the control signal $\overline{RD}$ suggests that it is a Read operation. Both signals are necessary to read from memory; the $\overline{MEMR}$ (Memory Read) control signal is generated by ANDing these two signals (see Section 4.14). The other status signals S_1 and S_0 are generally not needed in simple systems.
4. In the Memory Write cycle, the 8085 writes (stores) data in memory, using the control signal $\overline{WR}$ and the status signal $IO/\overline{M}$.
5. In the Memory Read cycle, the 8085 asserts the $\overline{RD}$ signal to enable memory, and then the addressed memory places data on the data bus; on the other hand, in the Memory Write cycle, the 8085 places the data byte on the data bus and then asserts the $\overline{WR}$ signal to write into the addressed memory.

6. The Memory Read and Write cycles consist of three T-states. The Memory Read and Write cycles will not be asserted simultaneously—the microprocessor cannot read and write at the same time.

4.3 MEMORY INTERFACING

Memory is an integral part of a microcomputer system, and in this chapter, our focus will be on how to interface a memory chip with the microprocessor. While executing a program, the microprocessor needs to access memory quite frequently to read instruction codes and data stored in memory; the interfacing circuit enables that access. Memory has certain signal requirements to write into and read from its registers. Similarly, the microprocessor initiates a set of signals when it wants to read from and write into memory. The interfacing process involves designing a circuit that will match the memory requirements with the microprocessor signals.

In the following sections, we will examine memory structure and its requirements and the 8085 Memory Read and Write machine cycles. Then we will derive the basic steps necessary to interface memory with the 8085. In the last chapter, we discussed a hypothetical memory chip and the concepts in addressing. In this chapter, we will illustrate memory interfacing, using memory chips such as 2732 EPROM and 6116 static R/W memory, and will discuss address decoding and memory addresses.

4.3.1 Memory Structure and Its Requirements

As discussed in Chapter 3, Read/Write memory (R/WM) is a group of registers to store binary information. Figure 4.11(a) shows a typical R/W memory chip; it has 2048 registers and each register can store eight bits indicated by eight input and eight output data lines.* The chip has 11 address lines A_{10}–A_0, one Chip Select ($\overline{CS}$), and two control lines: Read ($\overline{RD}$) to enable the output buffer and Write ($\overline{WR}$) to enable the input buffer. Figure 4.11(a) also shows the internal decoder to decode the address lines. Figure 4.11(b) shows the logic diagram of a typical EPROM (Erasable Programmable Read-Only Memory) with 4096 (4K) registers. It has 12 address lines A_{11}–A_0, one Chip Select ($\overline{CS}$), and one Read control signal. This chip must be programmed (written into) before it can be used as a read-only memory. Figure 4.11(b) also shows a quartz window on the chip that is used to expose the chip to ultraviolet rays for erasing the program. Once the chip is programmed, the window is covered with opaque tape to avoid accidental erasing. For interfacing the R/W memory, Figure 4.11(a), and the EPROM, Figure 4.11(b), the process is similar; the only difference is that the EPROM does not require the $\overline{WR}$ signal.

*In a typical memory chip, the input and output data lines are not shown separately, but are shown as a group of eight data lines.

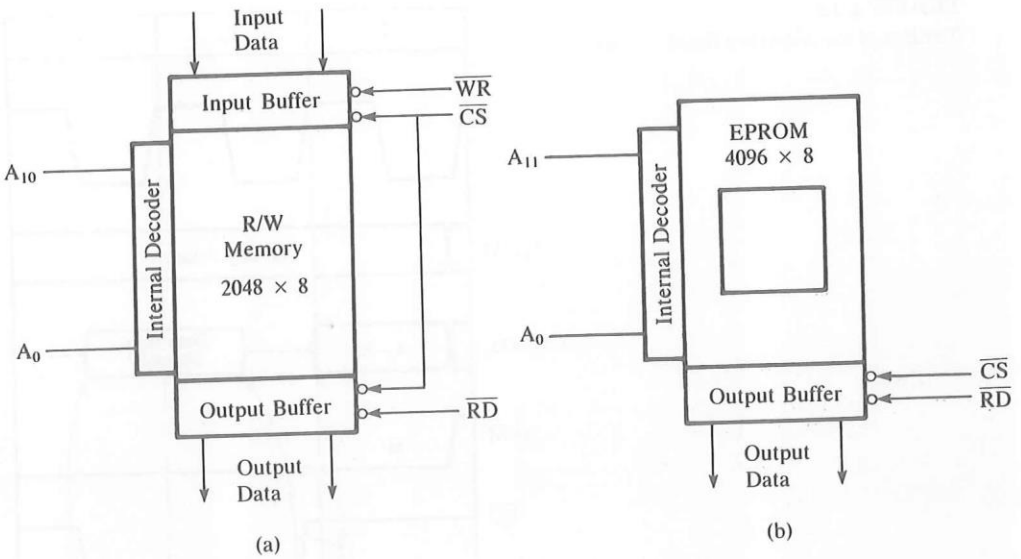


FIGURE 4.11
Typical Memory Chips: R/W Static Memory (a) and EPROM (b)

4.3.2 Basic Concepts in Memory Interfacing

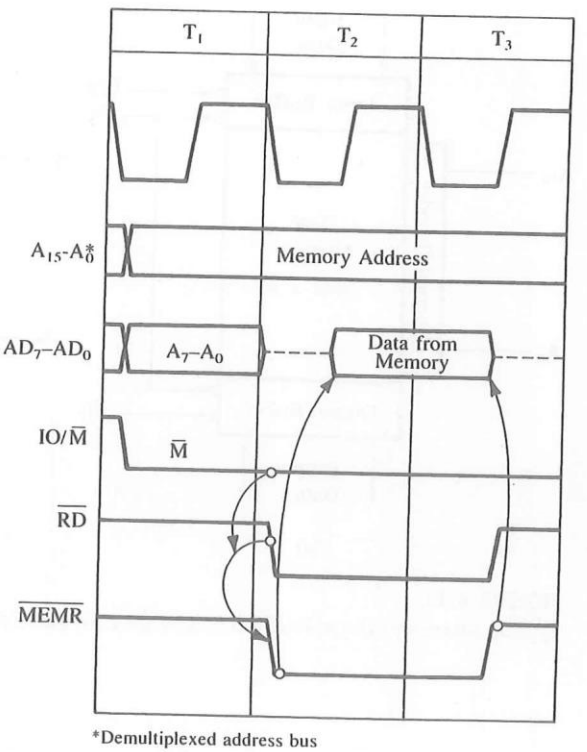
The primary function of memory interfacing is that the microprocessor should be able to read from and write into a given register of a memory chip. Recall from Chapter 3 that to perform these operations, the microprocessor should

1. be able to select the chip.
2. identify the register.
3. enable the appropriate buffer.

Let us examine the timing diagram of the Memory Read operation (Figure 4.12) to understand how the 8085 can read from memory. Figure 4.12 is the M_2 cycle of Figure 4.10 except that the address bus is demultiplexed. We could also use the M_1 cycle to illustrate these interfacing concepts.

1. The 8085 places a 16-bit address on the address bus, and with this address only one register should be selected. For the memory chip in Figure 4.11(a), only 11 address lines are required to identify 2048 registers. Therefore, we can connect the low-order address lines A_{10}–A_0 of the 8085 address bus to the memory chip. The internal decoder of the memory chip will identify and select the register for the EPROM, Figure 4.11(a).

FIGURE 4.12
Timing of the Memory Read Cycle

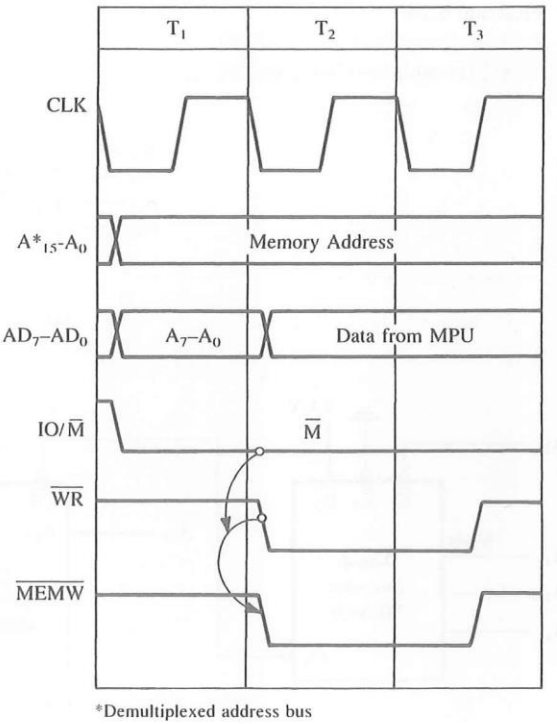


2. The remaining 8085 address lines (A_{15}–A_{11}) should be decoded to generate a Chip Select ($\overline{CS}$) signal unique to that combination of address logic (illustrated in Examples 4.3 and 4.4).
3. The 8085 provides two signals—IO/$\overline{M}$ and $\overline{RD}$—to indicate that it is a memory read operation. The IO/$\overline{M}$ and $\overline{RD}$ can be combined to generate the $\overline{MEMR}$ (Memory Read) control signal that can be used to enable the output buffer by connecting to the memory signal $\overline{RD}$.
4. Figure 4.12 also shows that memory places the data byte from the addressed register during T_2, and that is read by the microprocessor before the end of T_3.

To write into a register, the microprocessor performs similar steps as it reads from a register. Figure 4.13 shows the Memory Write cycle. In the Write operation, the 8085 places the address and data and asserts the IO/$\overline{M}$ signal. After allowing sufficient time for data to become stable, it asserts the Write ($\overline{WR}$) signal. The IO/$\overline{M}$ and $\overline{WR}$ signals can be combined to generate the $\overline{MEMW}$ control signal that enables the input buffer of the memory chip and stores the byte in the selected memory register.

To interface memory with the microprocessor, we can summarize the above steps as follows:

FIGURE 4.13
Timing of the Memory Write Cycle



1. Connect the required address lines of the address bus to the address lines of the memory chip.
2. Decode the remaining address lines of the address bus to generate the Chip Select signal, as discussed in the next section (4.3.3), and connect the signal to select the chip.
3. Generate control signals $\overline{\text{MEMR}}$ and $\overline{\text{MEMW}}$ by combining $\overline{\text{RD}}$ and $\overline{\text{WR}}$ signals with IO/$\overline{\text{M}}$, and use them to enable appropriate buffers.

4.3.3 Address Decoding

The process of address decoding should result in identifying a register for a given address. We should be able to generate a unique pulse for a given address. For example, in Figure 4.11(b), 12 address lines (A_{11}–A_0) are connected to the memory chip, and the remaining four address lines (A_{15}–A_{12}) of the 8085 microprocessor must be decoded. Figure 4.14 shows two methods of decoding these lines: one by using a NAND gate and the other by using a 3-to-8 decoder. The output of the NAND goes active and selects the chip only when all address lines A_{15}–A_{12} are at logic 1. We can obtain the same result by using O_7 of the 3-to-8 decoder, which is capable of decoding eight different input addresses. In the decoder circuit, three input lines can have eight different logic combinations from 000 to 111; each input combination can be identified by the corresponding output line if Enable

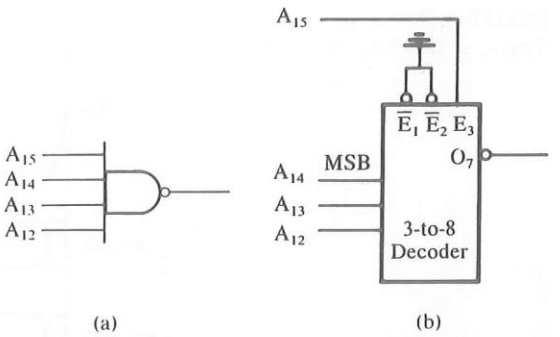


FIGURE 4.14
Address Decoding Using NAND Gate (a) and 3-to-8 Decoder (b)

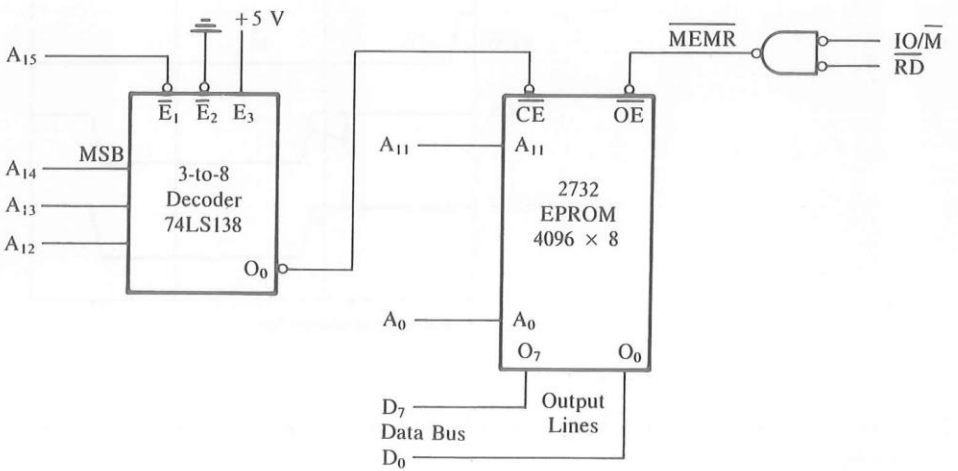


FIGURE 4.15
Interfacing the 2732 EPROM

lines are active. In this circuit, the Enable lines $\overline{E}_1$ and $\overline{E}_2$ are enabled by grounding, and A_{15} must be at logic 1 to enable E_3. We will use this address decoding scheme to interface a 4K EPROM and a 2K R/W memory as illustrated in the next two examples.

4.3.4 Interfacing Circuit

Figure 4.15 shows an interfacing circuit using a 3-to-8 decoder to interface the 2732 EPROM memory chip. It is assumed here that the chip has already been programmed, and we will analyze the interfacing circuit in terms of the same three steps outlined previously:

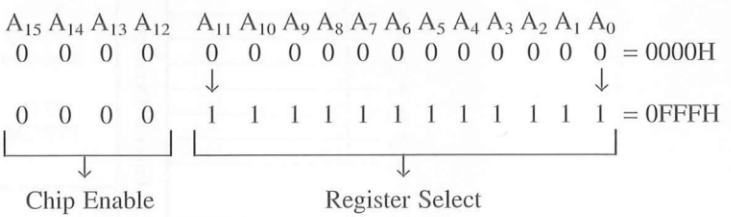
Step 1: The 8085 address lines A_{11}–A_0 are connected to pins A_{11}–A_0 of the memory chip to address 4096 registers.

Step 2: The decoder is used to decode four address lines A_{15}–A_{12}. The output O_0 of the decoder is connected to Chip Enable ($\overline{CE}$). The $\overline{CE}$ is asserted only when the address on A_{15}–A_{12} is 0000; A_{15} (low) enables the decoder and the input 000 asserts the output O_0.

Step 3: For this EPROM, we need one control signal: Memory Read ($\overline{MEMR}$), active low. The $\overline{MEMR}$ is connected to $\overline{OE}$ to enable the output buffer; $\overline{OE}$ is the same as $\overline{RD}$ in Figure 4.11.

4.35 Address Decoding and Memory Addresses

We can obtain the address range of this memory chip by analyzing the possible logic levels on the 16 address lines. The logic levels on the address lines A_{15}–A_{12} must be 0000 to assert the Chip Enable, and the address lines A_{11}–A_0 can assume any combinations from all 0s to all 1s. Therefore, the memory address of this chip ranges from 0000H to 0FFFH, as shown below.



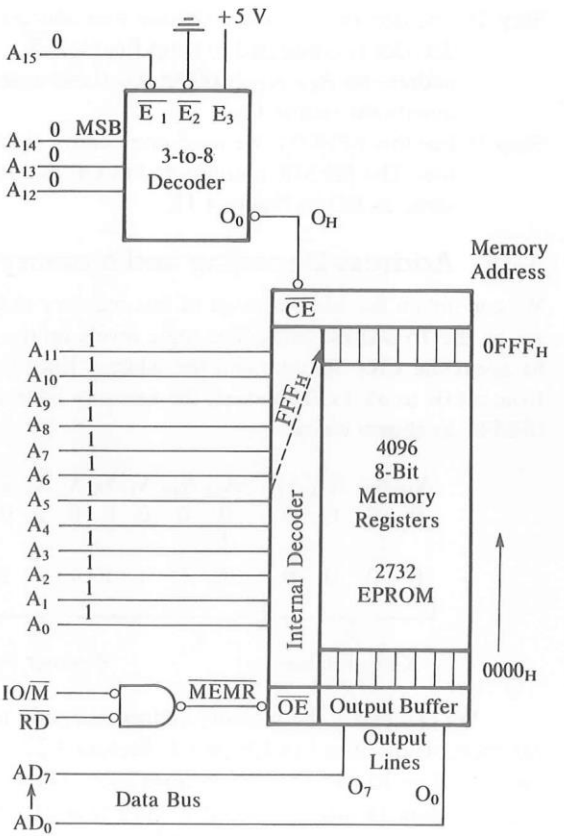
We can verify the memory address range in terms of our analogy of page and line numbers, as discussed in Chapter 3, Section 3.22. The chip's 4096 bytes of memory can be viewed as 16 pages with 256 lines each. The high-order Hex digits range from 00 to 0F, indicating 16 pages—0000H to 00FFH and 0100H to 01FFH, for example.

Now, to examine how an address is decoded and how the microprocessor reads from this memory, let us assume that the 8085 places the address 0FFFH on the address bus. The address 0000 (0H) goes to the decoder, and the output line O_0 of the decoder selects the chip. The remaining address FFFH goes on the address lines of the chip, and the internal decoder of the chip decodes the address and selects the register FFFH. Thus, the address 0FFFH selects the register as shown in Figure 4.16. When the 8085 asserts the $\overline{RD}$ signal, the output buffer is enabled and the contents of the register 0FFFH are placed on the data bus for the processor to read.

Analyze the interfacing circuit in Figure 4.17 and find its memory address range. **Example 4.5**

Solution

Figure 4.17 shows the interfacing of the 6116 memory chip with 2048 (2K) registers. The memory chip requires 11 address lines (A_{10}–A_0) to decode 2048 registers. The remaining address lines A_{15}–A_{11} are connected to the decoder. However, in this circuit, the decoder is enabled by the IO/$\overline{M}$ signal in addition to the address lines A_{15} and A_{14}, and the $\overline{RD}$ and



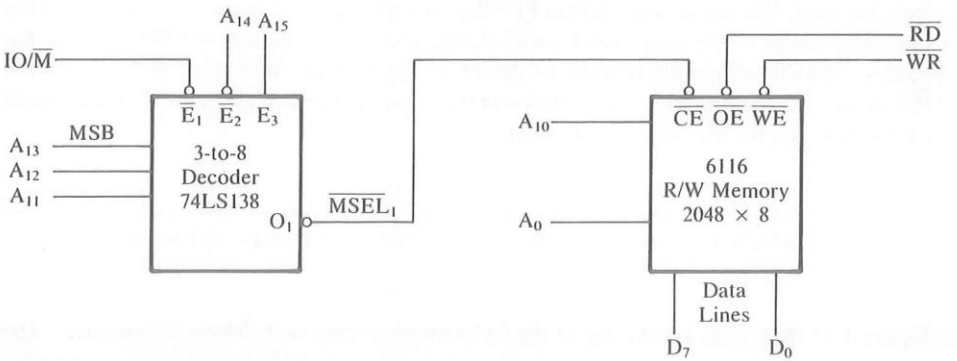


FIGURE 4.16
Address Decoding and Reading from Memory

FIGURE 4.17
Interfacing R/W Memory

to all 1s, as shown above.

4.4 INTERFACING THE 8155 MEMORY SEGMENT

The SDK-85 is a single-board microcomputer designed by Intel and widely used in college laboratories. The system is designed using the 8085 microprocessor and specially compatible devices, such as the 8155/8156.

The 8155/8156 includes multiple devices on the same chip. The 8155 has 256 bytes of R/W memory, two programmable I/O ports, and a timer. The 8156 is identical to the 8155, except that its Chip Enable (CE) signal is active high. The programmable I/O ports of this device are discussed in Chapter 14. The memory section of this chip and its memory addresses in the SDK-85 system will now be discussed.

4.4.1 Interfacing the 8155 Memory Section

Figure 4.18(a) shows the block diagram of the 8155 memory section. It has eight address lines, one $\overline{CE}$ (Chip Enable) line, and five lines compatible with the control and status signals of the 8085: IO/$\overline{M}$, ALE, $\overline{RD}$, $\overline{WR}$, and RESET. These control and status lines are not found in the general-purpose memory devices shown in the previous section. These lines eliminate the need for external demultiplexing of the bus AD_7–AD_0 and for generating separate control signals for memory and I/O.

Figure 4.18(b) also shows the internal structure of the 8155 memory section. The memory section includes 256×8 memory locations and an internal latch to demultiplex the bus lines AD_7–AD_0. The memory section also requires a Chip Enable ($\overline{CE}$) signal and the Memory Write ($\overline{MEMW}$) and Memory Read ($\overline{MEMR}$) control signals, generated internally by combining the IO/$\overline{M}$, $\overline{WR}$, and $\overline{RD}$ signals.

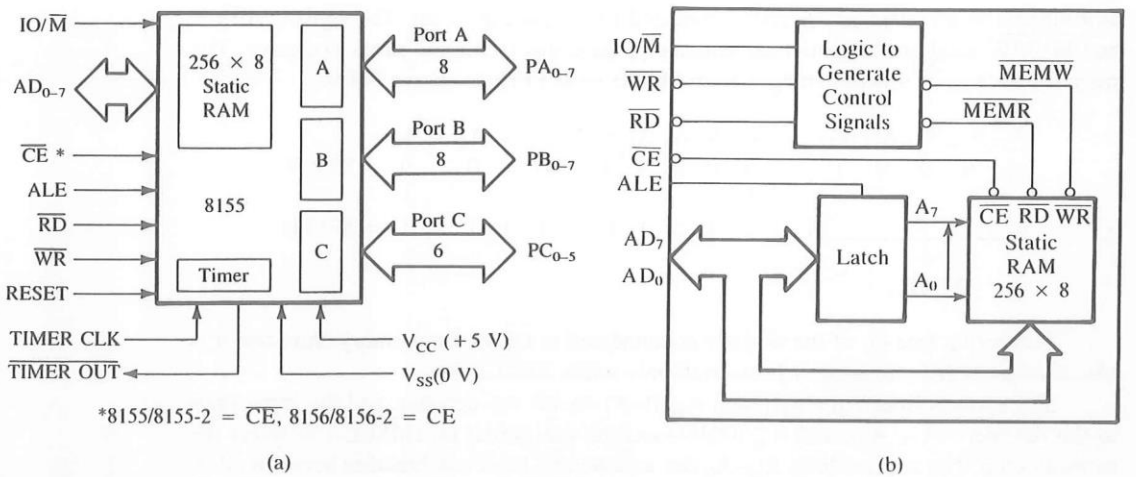


FIGURE 4.18
The 8155 Memory Section: The Block Diagram (a) and the Internal Structure (b)
SOURCE: (a) Intel Corporation, *Embedded Microprocessors* (Santa Clara, Calif.: Author, 1994), pp. 1-31.

FIGURE 4.19
Interfacing the 8155 Memory Schematic from the SDK-85 System
SOURCE: Intel Corporation, *SDK—85 User's Manual* (Santa Clara, Calif.: Author, 1978), Appendix B.

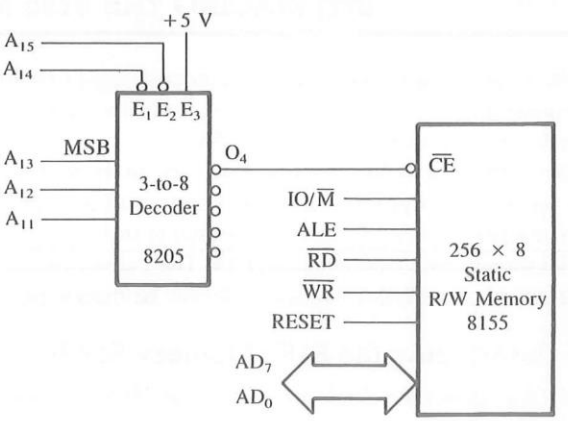


Figure 4.19 shows a schematic of the SDK-85 system of interfacing the 8155 memory section with the 8085. The 8205, a 3-to-8 decoder (identical to the 74LS138 decoder), decodes the address lines A_{15}–A_{11}, and the output line O_4 of the decoder enables the memory chip. The control and the status signals from the 8085 are connected directly to the respective signals on the memory chip. Similarly, the bus lines AD_7–AD_0 are also connected directly to the memory chip to address any one of the 256 memory locations.

Example 4.6

Explain the decoding logic and the memory address range of the 8155 shown in Figure 4.19.

Solution

The interfacing logic shows the 3-to-8 decoder; its output line 4 (O_4) is used to select the 8155. The address lines A_{11} to A_{13} are connected as input to the decoder, and the lines A_{15} and A_{14} are used as active low Enable lines. The third Enable line (active high) is permanently enabled by tying it to +5 V. The address lines A_{10}, A_9, and A_8 are not connected; thus, they are left as don't care lines. The output line O_4 of the decoder goes low when the address lines have the following address:

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	
0	0	1	0	0	0	0	0	= 20H (assuming the don't care lines are at logic 0)

The address lines AD_7–AD_0 can assume any combination of logic levels from all 0s to all 1s. Thus, the memory addresses of the 8155 memory will range from 2000H to 20FFH as follows.

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	= 2000H
0	0	1	0	0	0	0	0	1	1	1	1	1	1	1	1	= 20FFH
Chip Enable					Don't Care			Register Select								

In reality, the memory section of this 8155 uses the memory space from 2000H to 27FFH. In Figure 4.19, the address lines A_{10}, A_9, and A_8 are not connected; thus, they are don't care lines capable of assuming any logic state 0 or 1. Three don't care address lines can be assumed to have any one of the eight combinations from 000 to 111. Thus each combination can generate one set of complete addresses. The address range given by assuming all don't care lines at logic 0 is, by convention, specified as the memory address range of the system or the primary address; the remaining address ranges are known as either foldback memory or mirror memory. In this example, the primary address range is 2000H to 20FFH and the foldback memory range is 2100H to 27FFH as shown in Table 4.3.

The Don't Care column in Table 4.3 shows all the additional logic combinations of the address lines A_{10}–A_8 from 001 to 111, thus generating seven additional address ranges called foldback (or mirror) memory. In reality, these are the same memory registers as 2000H to 20FFH. Attempting to store an instruction in location 2100H (or 2200H or 2700H) is the same as entering the instruction in location 2000H. This results in assigning eight different addresses to the same memory register. To find each address range, we should assume only one combination of the don't care lines at a time; this is particularly important when the don't care lines are not in a particular sequence. (See Problem 34 at the end of the chapter.)

5

Interfacing I/O Devices

The I/O devices, such as keyboards and displays, are the ears and eyes of the MPUs; they are the communication channels to the "outside world." Data can enter (or exit) in groups of eight bits using the entire data bus; this is called the parallel I/O mode. The other method is the serial I/O, whereby one bit is transferred using one data line; typical examples include peripherals such as the CRT terminal and cassette tape. In this chapter, we will focus on interfacing I/O devices in the parallel mode and will discuss the serial mode in Chapter 16.

In the last chapter, we discussed memory interfacing. The 8085 microprocessor uses a 16-bit address bus for identifying and accessing memory registers. This results in a numbering scheme ranging from 0000H to FFFFH, also known as memory space. Similarly, the microprocessor needs to identify I/O devices with a binary number. These I/O devices can be interfaced using addresses (binary numbers) from the memory space; this is called memory-mapped I/O. Another option is to have a separate numbering (addressing) scheme for I/O devices. The 8085 microprocessor has a separate 8-bit addressing scheme (I/O space) for I/O devices; this is called peripheral-mapped I/O (or I/O-mapped I/O), and the I/O space ranges from 00H to FFH. In the 8085-based systems, I/O devices can be interfaced using both techniques: peripheral-mapped I/O and memory-mapped I/O. In peripheral-mapped I/O, a device is identified with an 8-bit address and enabled by I/O-related control signals. On the other hand, in memory-mapped I/O, a device is identified with a 16-bit address and enabled by memory-related control signals. The process of data transfer in both is identical. Each device is assigned a binary address, called a device address or port number, through its interfacing circuit. When the microprocessor executes a data transfer instruction for an I/O device, it places the appropriate address on the address bus, sends the control signals, enables the interfacing device, and transfers data. The interfacing device is like a gate for data bits, which is

opened by the MPU whenever it intends to transfer data. In peripheral-mapped I/O, data bytes are transferred by using IN/OUT instructions, and in memory-mapped I/O, data bytes are transferred by using memory-related (LDA, STA, etc.) data transfer instructions.

To grasp the essence of interfacing techniques, first we will examine the machine cycles of I/O instructions and find out the timings when I/O data are arriving on the data bus, and then we will latch (or catch) that information. We will derive the basic concepts in interfacing I/O devices from the machine cycles. Then we will illustrate these concepts by interfacing LEDs as an output device and switches as an input device. Specifically, the chapter deals with how the 8085 selects an I/O device, what hardware chips are necessary, what software instructions are used, and how data are transferred.

OBJECTIVES

- ☐ Illustrate the 8085 bus contents and control signals when OUT and IN instructions are executed.
- ☐ Recognize the device (port) address of a peripheral-mapped I/O by analyzing the associated logic circuit.
- ☐ Illustrate the 8085 bus contents and control signals when memory-related instructions (LDA, STA, etc.) are executed.
- ☐ Recognize the device (port) address of a memory-mapped I/O by analyzing the associated logic circuit.
- ☐ Explain the differences between the peripheral-mapped and memory-mapped I/O techniques.
- ☐ Interface an I/O device to a microcomputer for a specified device address by using logic gates and MSI chips, such as decoders, latches, and buffers.

5.1 BASIC INTERFACING CONCEPTS

The approach to designing an interfacing circuit for an I/O device is determined primarily by the instructions to be used for data transfer. An I/O device can be interfaced with the 8085 microprocessor either as a peripheral I/O or as a memory-mapped I/O. In the peripheral I/O, the instructions IN/OUT are used for data transfer, and the device is identified by an 8-bit address. In the memory-mapped I/O, memory-related instructions are used for data transfer, and the device is identified by a 16-bit address. However, the basic concepts in interfacing I/O devices are similar in both methods. Peripheral I/O is described in the following section, and memory-mapped I/O is described in Section 5.4.

5.1.1 Peripheral I/O Instructions

The 8085 microprocessor has two instructions for data transfer between the processor and the I/O device: IN and OUT. The instruction IN (Code DB) inputs data from an input device (such as a keyboard) into the accumulator, and the instruction OUT (Code D3) sends the contents of the accumulator to an output device such as an LED display. These are 2-byte instructions, with the second byte specifying the address or the port number of an I/O device. For example, the OUT instruction is described as follows.

Opcode	Operand	Description
OUT	8-bit Port Address:	This is a two-byte instruction with the hexadecimal opcode D3, and the second byte is the port address of an output device. This instruction transfers (copies) data from the accumulator to the output device.

Typically, to display the contents of the accumulator at an output device (such as LEDs) with the address, for example, 01H, the instruction will be written and stored in memory as follows:

Memory Address	Machine Code	Mnemonics		Memory Contents
2050	D3	OUT 01H	; 2050 →	1 1 0 1 0 0 1 1 = D3H
2051	01		; 2051 →	0 0 0 0 0 0 0 1 = 01H

(Note: The memory locations 2050H and 2051H are chosen here arbitrarily for the illustration.)

If the output port with the address 01H is designed as an LED display, the instruction OUT will display the contents of the accumulator at the port. The second byte of this OUT instruction can be any of the 256 combinations of eight bits, from 00H to FFH. Therefore, the 8085 can communicate with 256 different output ports with device addresses ranging from 00H to FFH. Similarly, the instruction IN can be used to accept data from 256 different input ports. Now the question remains: How does one assign a device address or a port number to an I/O device from among 256 combinations? The decision is arbitrary and somewhat dependent on available logic chips. To understand a device address, it is necessary to examine how the microprocessor executes IN/OUT instructions.

5.1.2 I/O Execution

The execution of I/O instructions can best be illustrated using the example of the OUT instruction given in the previous section (5.1.1). The 8085 executes the OUT instruction in three machine cycles, and it takes ten T-states (clock periods) to complete the execution.

OUT INSTRUCTION (8085)

In the first machine cycle, M_1 (Opcode Fetch, Figure 5.1), the 8085 places the high-order memory address 20H on A_{15}–A_8 and the low-order address 50H on AD_7–AD_0. At the same time, ALE goes high and IO/$\overline{M}$ goes low. The ALE signal indicates the availability of the address on AD_7–AD_0, and it can be used to demultiplex the bus. The IO/$\overline{M}$, being low, indicates that it is a memory-related operation. At T_2, the microprocessor sends the $\overline{RD}$ control signal, which is combined with IO/$\overline{M}$ (externally, see Chapter 4) to generate the $\overline{MEMR}$ signal, and the processor fetches the instruction code D3 using the data bus.

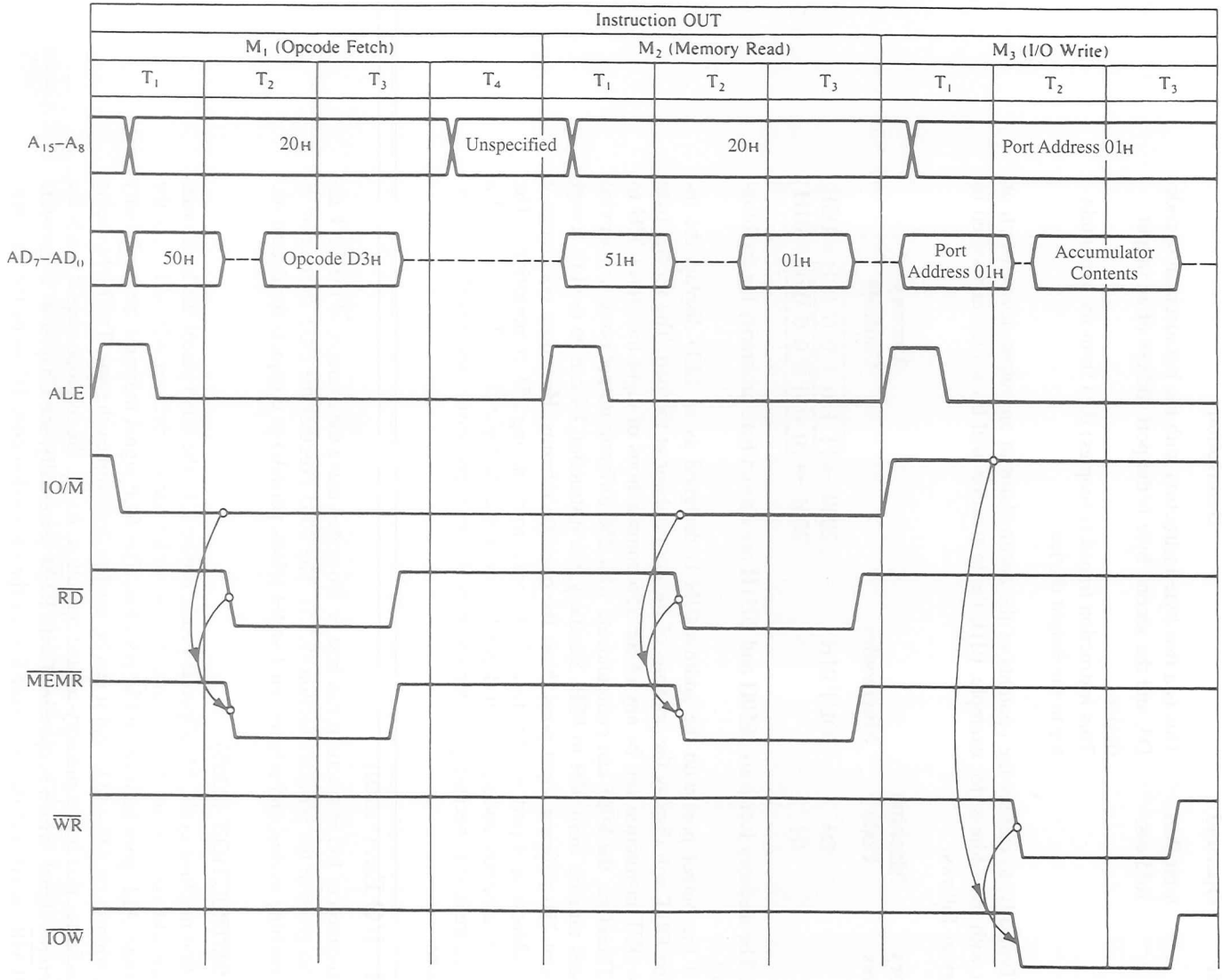


FIGURE 5.1
8085 Timing for Execution of OUT Instruction

When the 8085 decodes the machine code D3, it finds out that the instruction is a 2-byte instruction and that it must read the second byte.

In the second machine cycle, M_2 (Memory Read), the 8085 places the next address, 2051H, on the address bus and gets the device address 01H via the data bus.

In the third machine cycle, M_3 (I/O Write), the 8085 places the device address 01H on the low-order (AD_7–AD_0) as well as the high-order (A_{15}–A_8) address bus. The $\overline{IO/M}$ signal goes high to indicate that it is an I/O operation. At T_2, the accumulator contents are placed on the data bus (AD_7–AD_0), followed by the control signal $\overline{WR}$. By ANDing the $\overline{IO/M}$ and $\overline{WR}$ signals, the $\overline{IOW}$ (see Figure 4.5) signal can be generated to enable an output device.

Figure 5.1 shows the execution timing of the OUT instruction. The information necessary for interfacing an output device is available during T_2 and T_3 of the M_3 cycle. The data byte to be displayed is on the data bus, the 8-bit device address is available on the low-order as well as high-order address bus, and availability of the data byte is indicated by the $\overline{WR}$ control signal. The availability of the device address on both segments of the address bus is redundant information; in peripheral I/O, only one segment of the address bus (low or high) is sufficient for interfacing. The data byte remains on the data bus only for two T-states, then the processor goes on to execute the next instruction. Therefore, the data byte must be latched now, before it is lost, using the device address and the control signal (Section 5.13).

IN INSTRUCTION

The 8085 instruction set includes the instruction IN to read (copy) data from input devices such as switches, keyboards, and A/D data converters. This is a two-byte instruction that reads an input device and places the data in the accumulator. The first byte is the opcode, and the second byte specifies the port address. Thus, the addresses for input devices can range from 00H to FFH. The instruction is described as

IN 8-bit This is a two-byte instruction with the hexadecimal opcode DB, and the second byte is the port address of an input device.

This instruction reads (copies) data from an input device and places the data byte in the accumulator.

To read switch positions, for example, from an input port with the address 84H, the instructions will be written and stored in memory as follows:

Memory Address	Machine Code	Mnemonics		Memory Contents
2065	DB	IN 84H	; 2065 →	1 1 0 1 1 0 1 1 = DBH
2066	84		; 2066 →	1 0 0 0 0 1 0 0 = 84H

(Note: The memory locations 2065H and 66H are selected arbitrarily for the illustration.)

When the microprocessor is asked to execute this instruction, it will first read the machine codes (or bytes) stored at locations 2065H and 2066H, then read the switch po-

sitions at port 84H by enabling the interfacing device of the port. The data byte indicating switch positions from the input port will be placed in the accumulator. Figure 5.2 shows the timing of the IN instruction; M_1 and M_2 cycles are identical to that of the OUT instruction. In the M_3 cycle, the 8085 microprocessor places the address of the input port (84H) on the low-order address bus AD_7–AD_0 as well as on the high-order address bus A_{15}–A_8 and asserts the $\overline{RD}$ signal, which is used to generate the I/O Read ($\overline{IOR}$) signal. The $\overline{IOR}$ enables the input port, and the data from the input port are placed on the data bus and transferred into the accumulator.

Machine cycle M_3 (Figure 5.2) is similar to the M_3 cycle of the OUT instruction; the only differences are (1) the control signal is $\overline{RD}$ instead of $\overline{WR}$, and (2) data flow from an input port to the accumulator rather than from the accumulator to an output port.

5.1.3 Device Selection and Data Transfer

The objective of interfacing an output device is to get information or a result out of the processor and store it or display it. The OUT instruction serves that purpose; during the M_3 cycle of the OUT instruction the processor places that information (accumulator contents) on the data bus. If we connect the data bus to a latch, we can catch that information and display it via LEDs or a printer. Now the questions are: (1) When should we enable the latch to catch that information? and (2) What should be the address of that latch? The answers to both questions can be found in the M_3 cycle (Figure 5.1). The latch should be enabled when IO/$\overline{M}$ is high and $\overline{WR}$ is active low. Similarly, the address of an output port is also on the address bus during M_3 (it is 01H in Figure 5.1). Now the task is to generate one pulse by decoding the address bus (A_7–A_0 or A_{15}–A_8) to indicate the presence of the port address we are interested in, generate a timing pulse by combining IO/$\overline{M}$ and WR signals to indicate that the data byte we are looking for is on the data bus, and use these pulses (by combining them) to enable the latch. These steps are summarized as follows. (For all subsequent discussion, the bus A_7–A_0 is assumed to be the demultiplexed bus AD_7–AD_0.)

1. Decode the address bus to generate a unique pulse corresponding to the device address on the bus; this is called the **device address pulse** or **I/O address pulse.**
2. Combine (AND) the device address pulse with the control signal to generate a device select (I/O select) pulse that is generated only when both signals are asserted.
3. Use the I/O select pulse to activate the interfacing device (I/O port).

The block diagram (Figure 5.3) illustrates these steps for interfacing an I/O device. In Figure 5.3, address lines A_7–A_0 are connected to a decoder, which will generate a unique pulse corresponding to each address on the address lines. This pulse is combined with the control signal to generate a device select pulse, which is used to enable an output latch or an input buffer.

Figure 5.4 shows a practical decoding circuit for the output device with address 01H. Address lines A_7–A_0 are connected to the 8-input NAND gate that functions as a decoder. Line A_0 is connected directly, and lines A_7–A_1 are connected through the inverters. When the address bus carries address 01H, gate G_1 generates a low pulse; otherwise, the

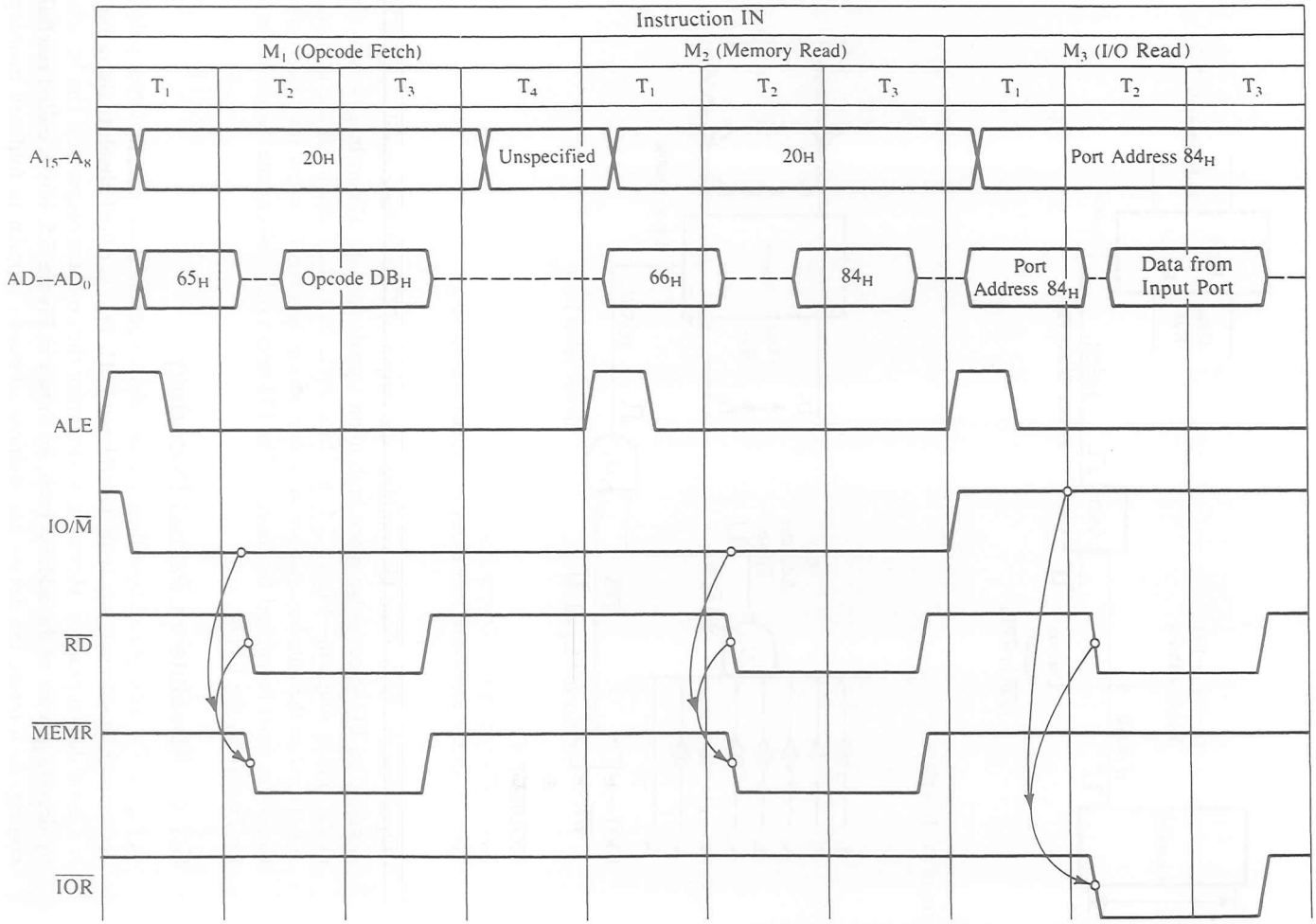


FIGURE 5.2
8085 Timing for Execution of IN Instruction