

Chapter-2

Microprocessor Architecture and Microcomputer System

Microprocessor

- ❑ A microprocessor is programmable logic device, designed with registers, flip-flops, and timing elements. It is a single chip made by LSI or VLSI technology. It is the CPU of a microcomputer.

Microprocessor Unit (MPU)

- ❑ In early processors the different units of a CPU are made not in a single chip rather in different separate chips. These group of chips are called a Micro Processor Unit (MPU).

2.1 MICROPROCESSOR ARCHITECTURE AND ITS OPERATIONS

Architecture of Microprocessor

- The internal logic diagram of a microprocessor is called its architecture.

Different Categories of Functions Performed by a Microprocessor

All the various functions performed by the microprocessor can be classified in three categories:

- Microprocessor-initiated operations.
- Internal data operations.
- Peripheral (or externally) initiated operations.

2.1.1 Microprocessor-Initiated Operations

The MPU performs primarily four operations:

- Memory Read: Reads data from memory.
- Memory Write: Writes data into memory.
- I/O Read: Accepts data from input devices.
- I/O Write: Sends data to output devices.

Steps to Communicate with Peripherals or Memory Location

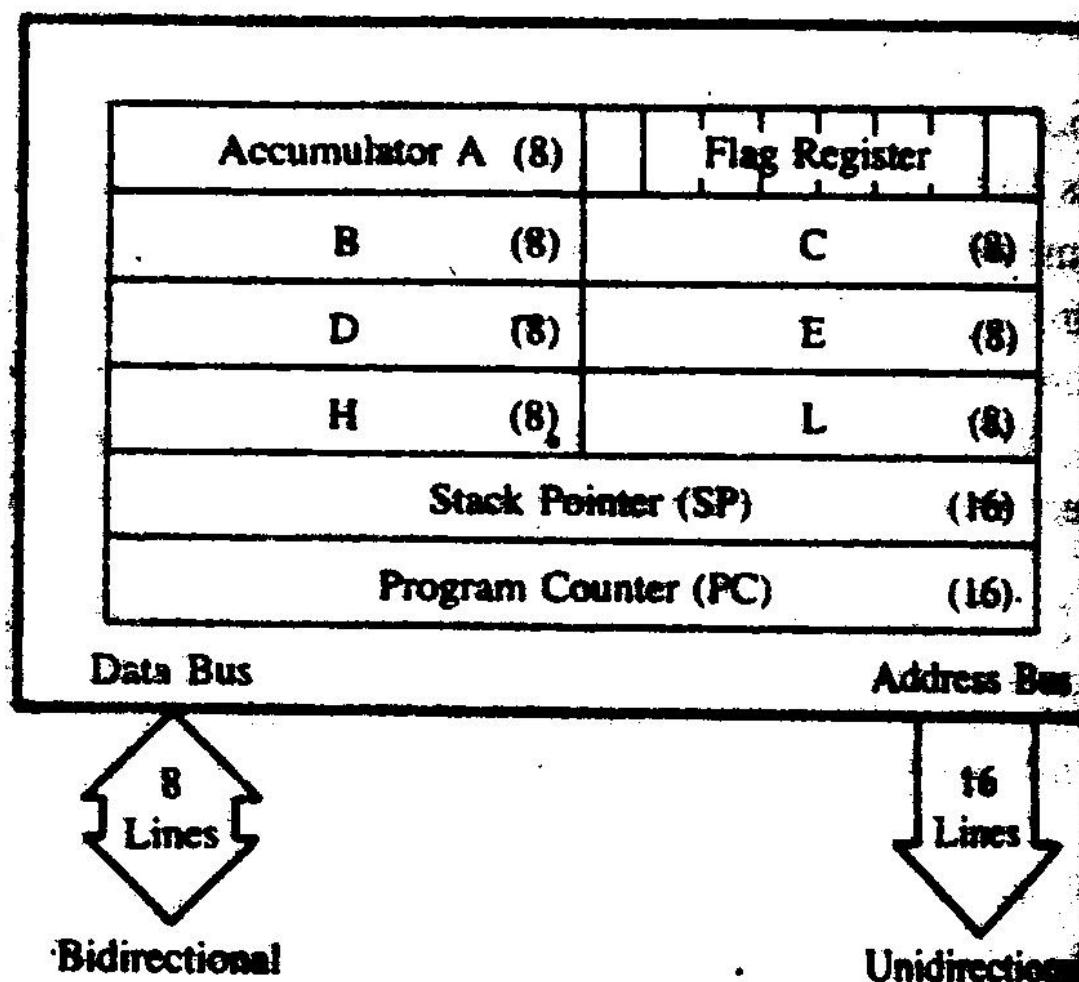
To communicate with a peripheral or a memory location, the MPU needs to perform the following steps:

- Step 1: Identify the peripheral or the memory location (with its address)
- Step 2: Transfer data.
- Step 3: Provide timing or synchronization signals.

2.1.2 Internal Data Operations and the 8085/8080A Registers

The internal architecture of the 8085/8080A microprocessor determines how and what operations can be performed with data. These operations are:

- Store 8-bit data
- Perform arithmetic and logical operations.
- Test for conditions.
- Sequence the execution of instructions.
- Store data temporarily during execution in the defined R/W memory locations called the stack.

FIGURE 2.3**The 8085 Programmable Registers**

2.1.3 Peripheral or Externally Initiated Operations

External devices (or signals) can initiate the following operations for which individual pins on the microprocessor chip are assigned:

- Reset:** When all internal operations are suspended and the program counter(PC) is cleared (0000H).
- Interrupt:** Microprocessor can be interrupted from the normal execution and asked to execute some other instructions called service routine. After completing service routine it resumes its operation.
- Ready:** If the signal of 8085/8080A READY pin is low, the microprocessor enters into a Wait state. The pin is used to synchronize with slower peripherals with microprocessor.
- Hold:** When the HOLD is activated by an external signal, the microprocessor relinquishes control of system bus for external peripheral to use them. For example, HOLD signal is used in DMA data transfer.

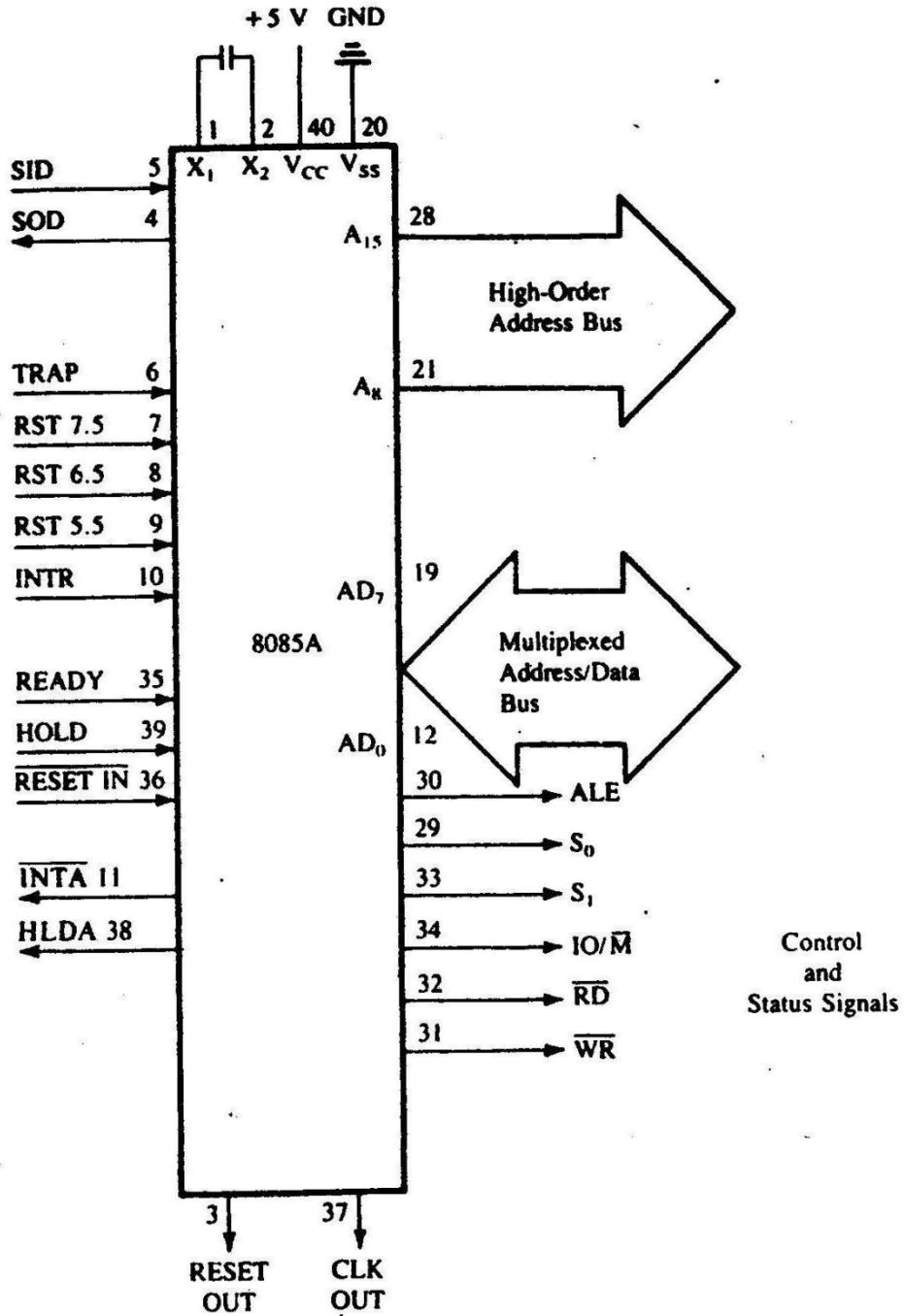
X ₁	1	40	V _{CC}
X ₂	2	39	HOLD
RESET OUT	3	38	HLDA
SOD	4	37	CLK (OUT)
SID	5	36	RESETIN
TRAP	6	35	READY
RST 7.5	7	34	IO/M
RST 6.5	8	33	S ₁
RST 5.5	9	32	RD
INTR	10	8085A	WR
INTA	11	31	ALE
AD ₀	12	29	S ₀
AD ₁	13	28	A ₁₅
AD ₂	14	27	A ₁₄
AD ₃	15	26	A ₁₃
AD ₄	16	25	A ₁₂
AD ₅	17	24	A ₁₁
AD ₆	18	23	A ₁₀
AD ₇	19	22	A ₉
V _{SS}	20	21	A ₈

8085 Pinout

Serial I/O Ports

Externally Initiated Signals

External Signal Acknowledgment



Control and Status Signals

Chapter-3

8085

Microprocessor Architecture and Memory Interfacing

3.1 THE 8085 MPU

Microprocessor Unit (MPU)

- The term microprocessor unit (MPU) define as 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.

Using above 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.

3.11 The 8085 Microprocessor

- The 8085A commonly known as the 8085
- It is an 8-bit general-purpose microprocessor
- It is capable of addressing 64K of memory.
- It has 40 pins and requires +5V power supply.
- It can operate with a 3-MHz clock frequency.
- It is an enhanced version of its predecessor 8080A.

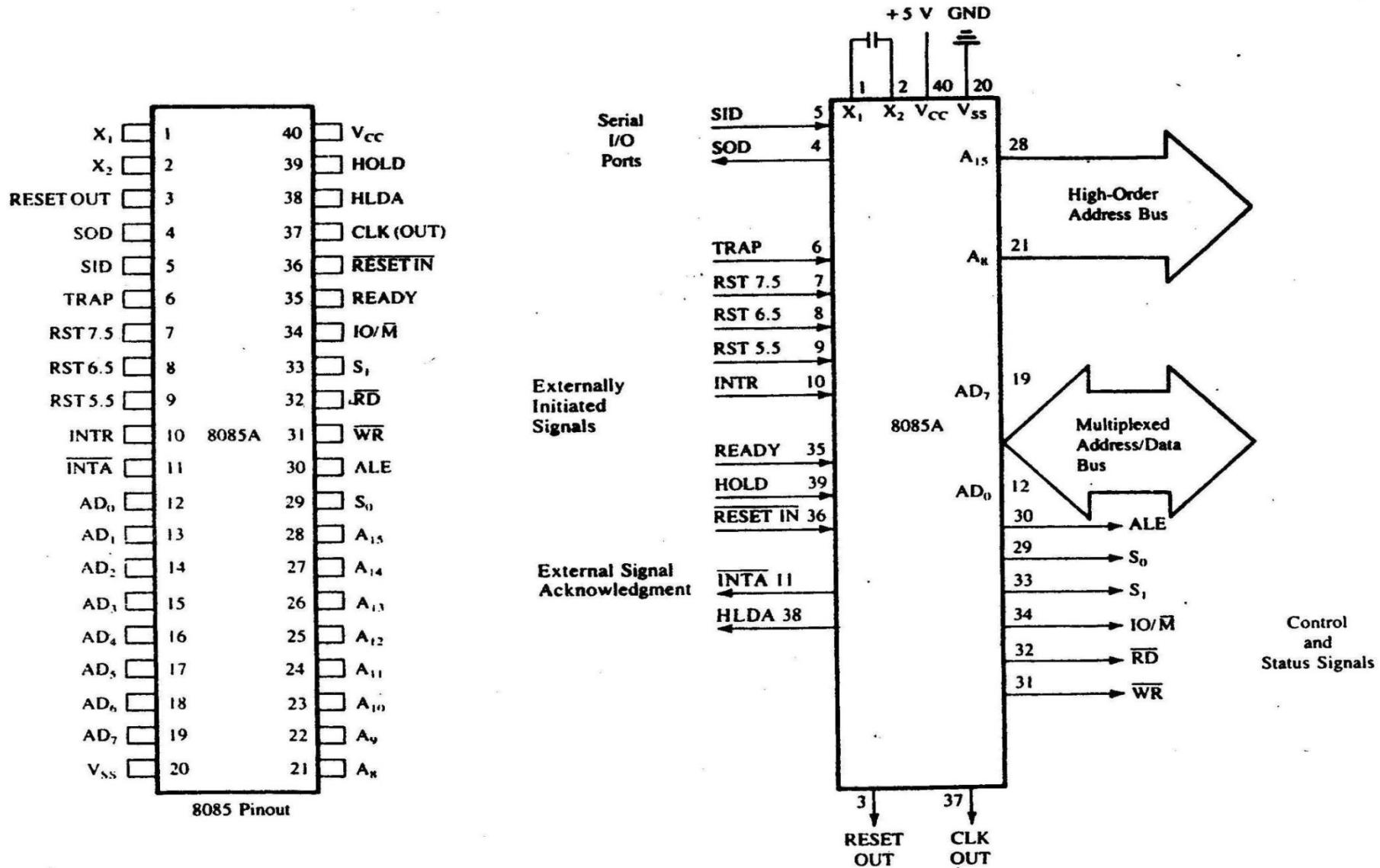


FIGURE 3.1
The 8085 Microprocessor Pinout and Signals

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

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

Figure 3.1 shows the logic pinout of the 8085 microprocessor. All the signal 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.
- 6) Serial I/O ports.

ADDRESS BUS

- A₁₅ - A₈ are unidirectional and high order address bus.

MULTIPLEXED ADDRESS/DATA BUS

- AD₇ - AD₀ are bidirectional and serve a dual purpose. During instruction fetch these are used as address lines and during operand fetch these are used as data lines.

CONTROL AND STATUS SIGNALS

Control and status signals indicate the nature of the operation.

- Control Signals: RD and WR.
 - Status Signals: IO/M, S_1 , and S_0 .
 - Special Signal: ALE. Indicate the beginning of the operation.
-
- RD: Low for read operation(active low).
 - WR: Low for write operation (active low).
 - IO/M: High for I/O operation (active high) and low for memory operation (active low).
 - S_1 , and S_0 : Similar to IO/M and rarely used in small system.

ALE (Address Latch Enable)

- This is a positive going pulse and generated every time 8085 begins an operation; it indicates that bits on AD₇ - AD₀ are address bits. This signal are used to latch the low-order address bus an generate a separate set of eight address lines; A₇ - A₀

EXTERNALLY INITIATED SIGNALS INCLUDING INTERRUPT

TABLE 3.2

8085 Interrupts and Externally Initiated Signals

- INTR (Input)
- INTA (Output)
- RST 7.5 (Inputs)
- RST 6.5
- RST 5.5
- TRAP (Input)
- HOLD (Input)
- HLDA (Output)
- READY (Input)

3.12 Microprocessor Communication and Bus Timing

Example: The instruction code 01001111 (4FH-MOV C,A) is stored in memory location 2005H. Illustrate the data flow and list the sequence of events when the instruction code is fetched by the MPU.

Solution: To fetch the byte, the MPU performs the following steps:

Step 1: The program counter places the 16-bit memory address on the address bus (Figure 3.2).

Figure 3.3 shows that at T_1 , the high-order address (20H) is placed on $A_{15} - A_8$ and the low-order address (05H) is placed on $AD_7 - AD_0$, ALE goes high, IO/M goes low.

Step 2: The control unit sends the control signal RD to enable the memory chip (Figure 3.2).

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

8085 MICROPROCESSOR ARCHITECTURE AND MEMORY INTERFACING

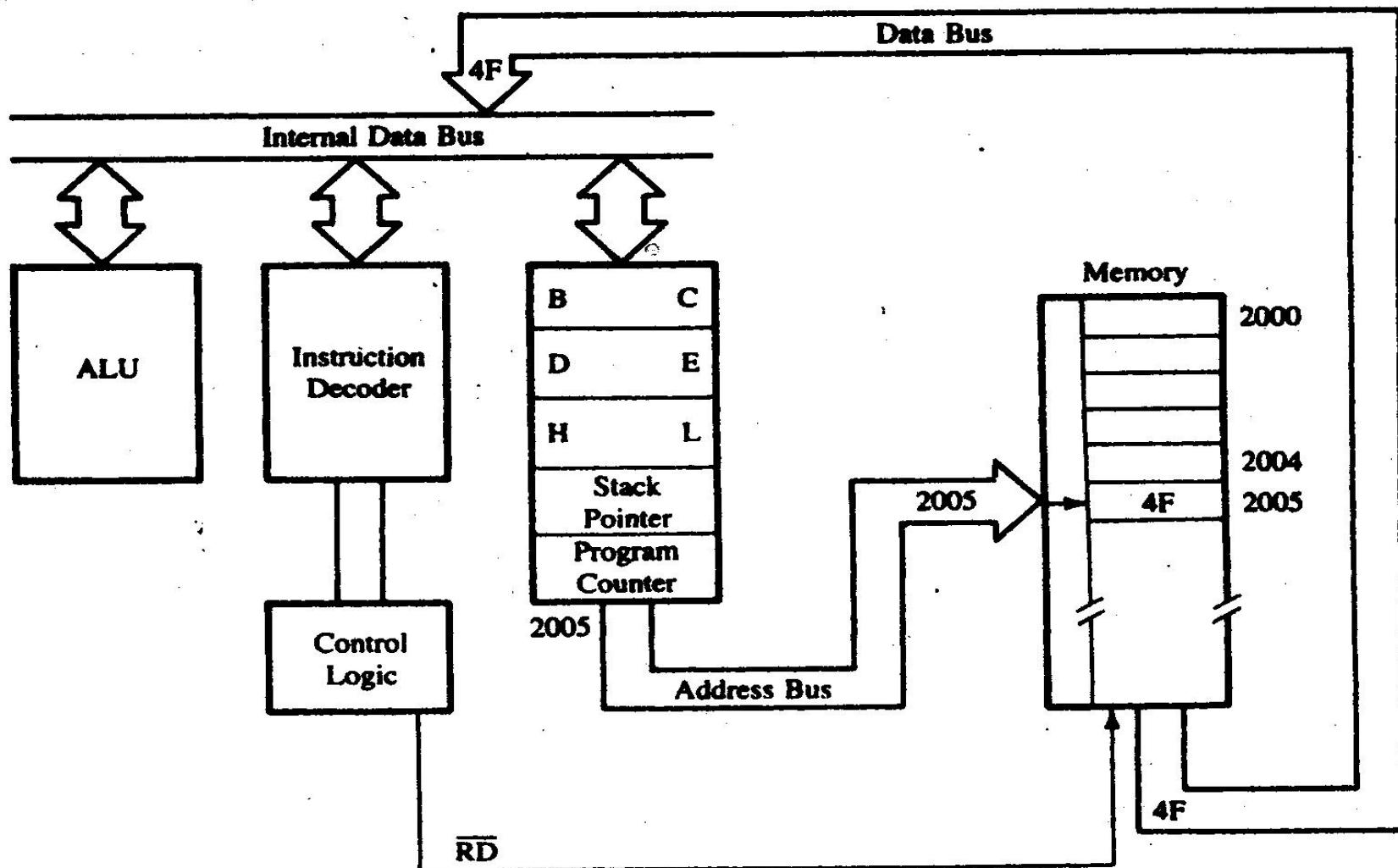


FIGURE 3.2
Data Flow from Memory to the MPU

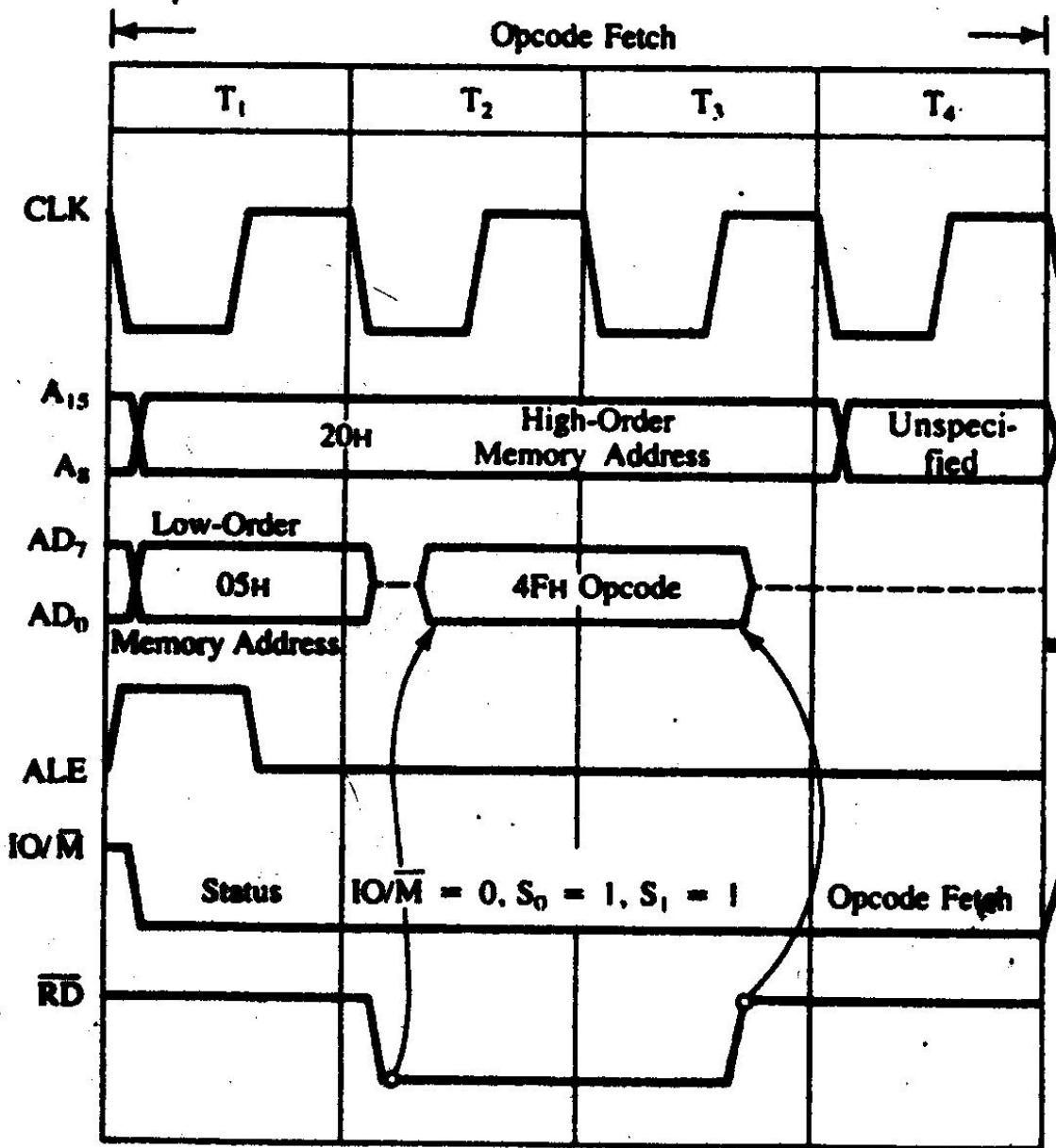


FIGURE 3.3

Timing: Transfer of Byte from Memory to MPU

The RD signal causes 4FH to be placed on bus AD₇ - AD₀ (shown by arrow).

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

3.13 Demultiplexing the Bus AD₇-AD₀

Figure 3.3 shows that the address on the high-order bus (20H) remains on the bus for three clock periods (T₁ to T₃). However, the low-order address (05H) is lost after the first clock period. The address needs to be latched for identifying the memory address. Otherwise, the address 2005H will change to 204FH after the first clock period.

Figure 3.4 shows that the uses of a latch (74LS373) and the ALE signal demultiplexes the low-order bus.

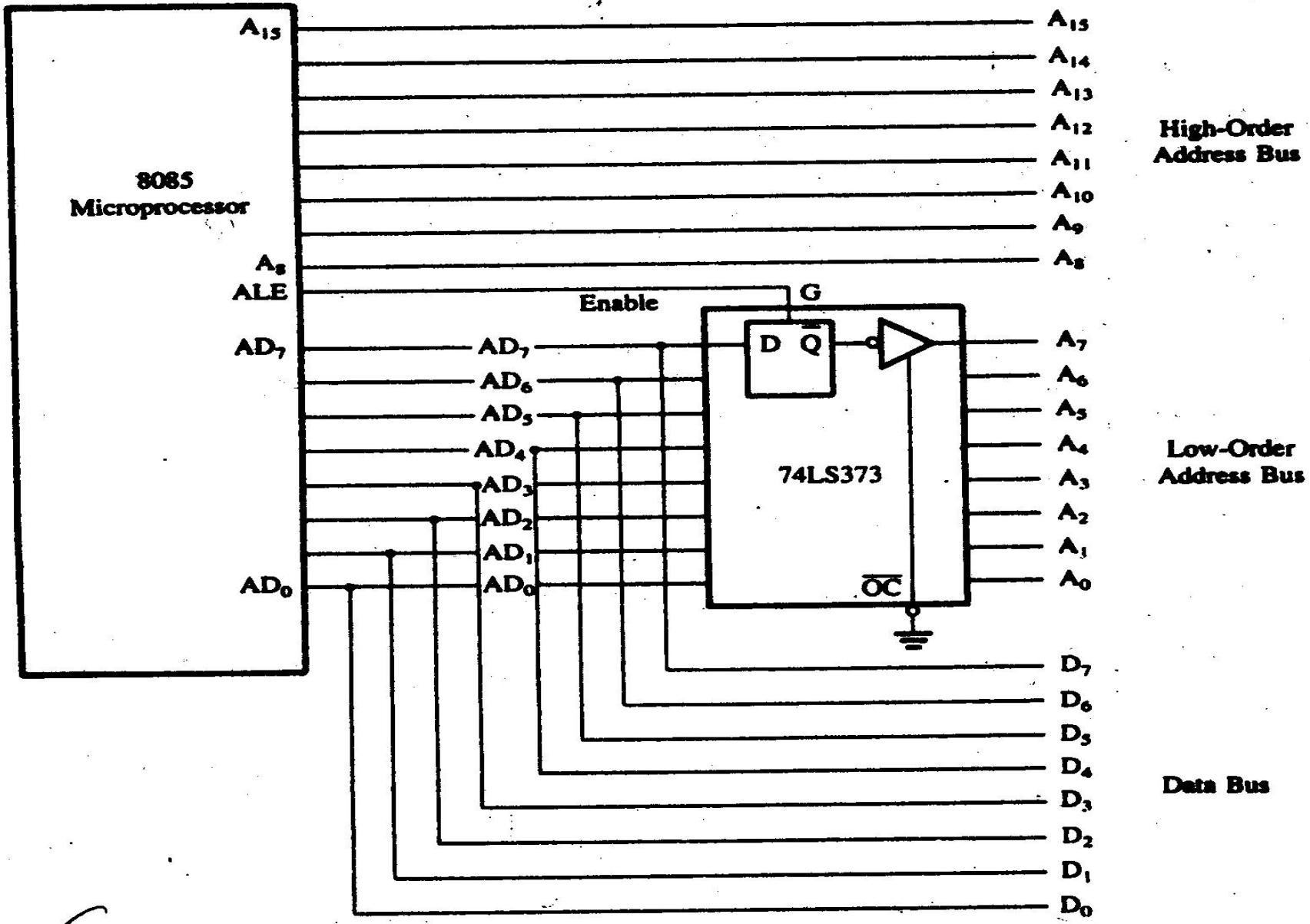


FIGURE 3.4
Schematic of Latching Low-Order Address Bus

Figure 3.3 shows that when the ALE is high (at T_1), the latch is transparent; this means the output changes according to the input data (05H).

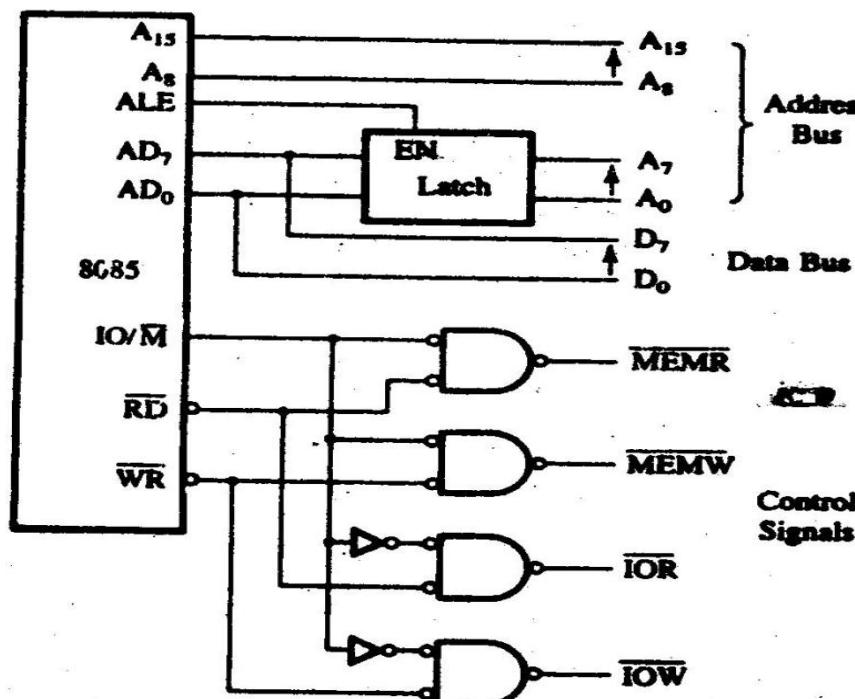
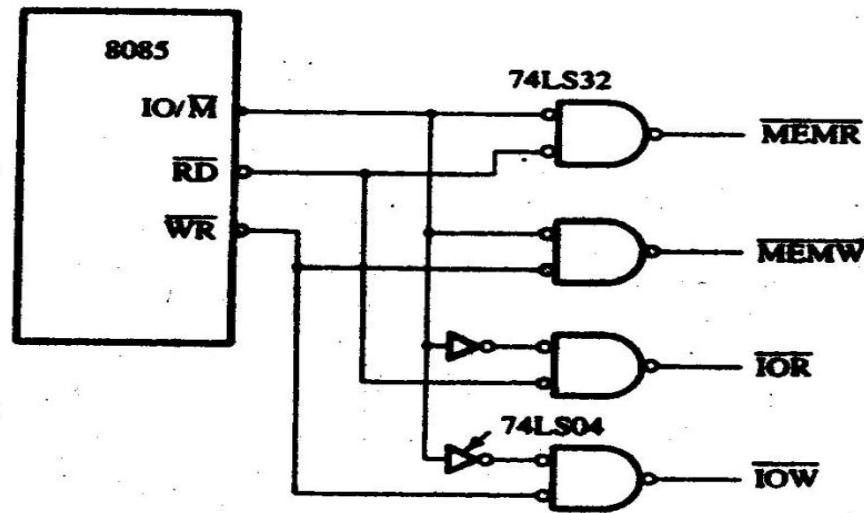
When the ALE goes low, the data 05H is latched until the ALE becomes high again.

3.14 Generating Control Signals

Figure 3.5 shows that four control signals are generated by combining $\overline{\text{RD}}$, $\overline{\text{WR}}$, and IO/M .

FIGURE 3.5

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

**FIGURE 3.6**

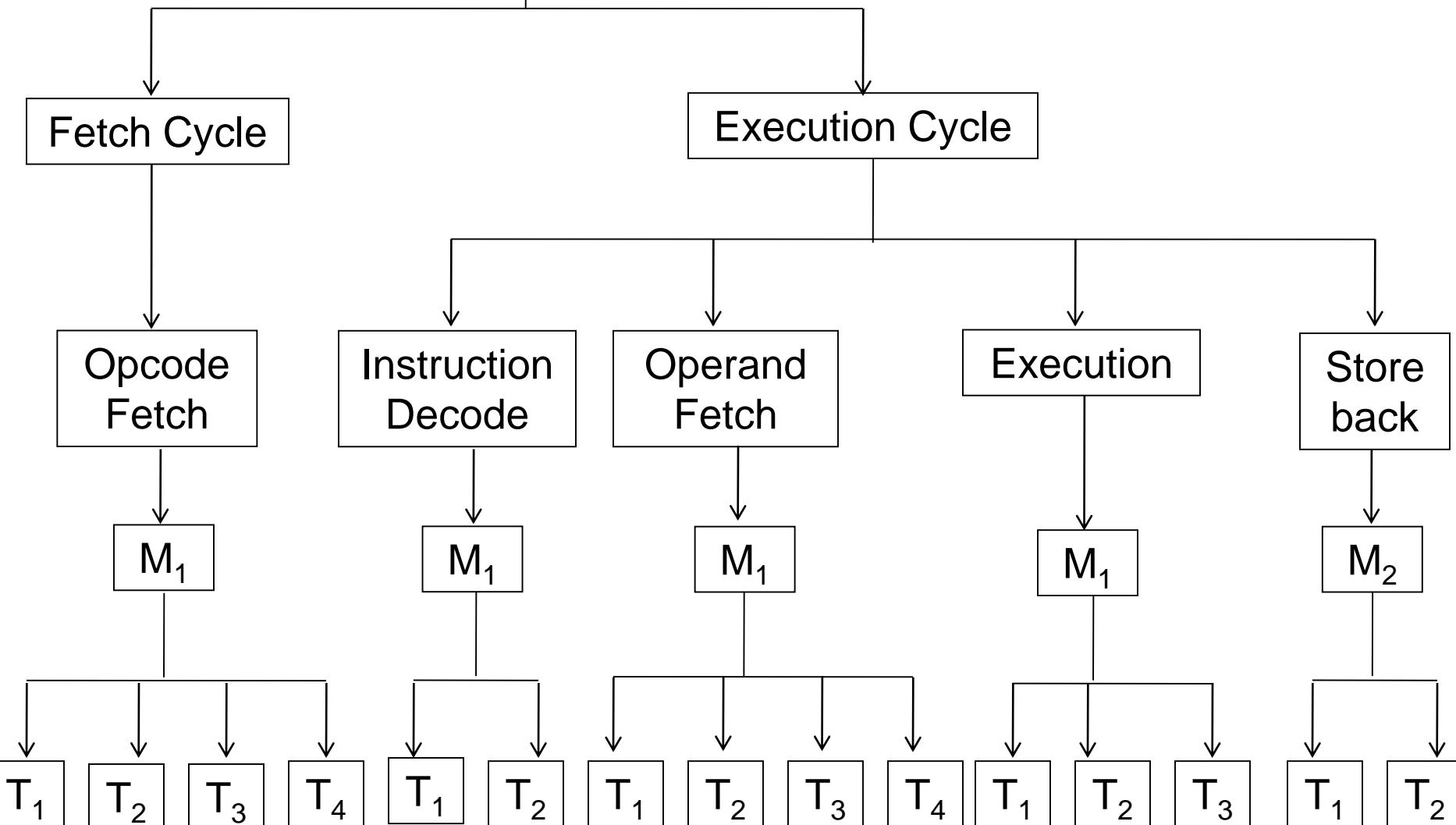
8085 Demultiplexed Address and Data Bus with Control Signals

3.21 The 8085 Machine Cycles and Bus Timing

To execute an instruction, 8085 needs to complete an instruction cycle. An instruction cycle is consisted of one or more machine cycles. A machine cycle is again consisted of two or more clock periods (T-states).

- Instruction cycle = Fetch cycle + Execution cycle
- Fetch cycle = Opcode fetch (from memory)
- Execution cycle = Opcode decode (in CPU) + Operand fetch [if necessary] (from memory) + Execution (in CPU) + Store back [optional] (to memory)

Instruction Cycle



3.22 Opcode Fetch Machine Cycle

To execute any instruction, 8085 at first fetches the opcode of the instruction and it requires a single machine cycle. In Figure 3.3 we have seen how the opcode 4FH (MOV C,A) is fetched from memory location 2005H.

3.23 Memory Read Machine Cycle (Operands-Data/Address)

In Figure 3.3 the instruction MOV C,A is a 1-byte instruction and only the opcode is fetched from memory, no operand (data or address) is read from memory. For an instruction which is more than 1-byte a further memory read operation is needed for operand(s) [data or address].

Example: The machine codes-00111110 (3EH-MVI A) and 00110010 (32H) – are stored in memory locations: 2000H and 2001H.

<u>Memory Location</u>	<u>Machine Code</u>	<u>Hex</u>	<u>Instruction</u>
2000H	00111110	3EH	MVI A,
2001H	00110010	32H	32H

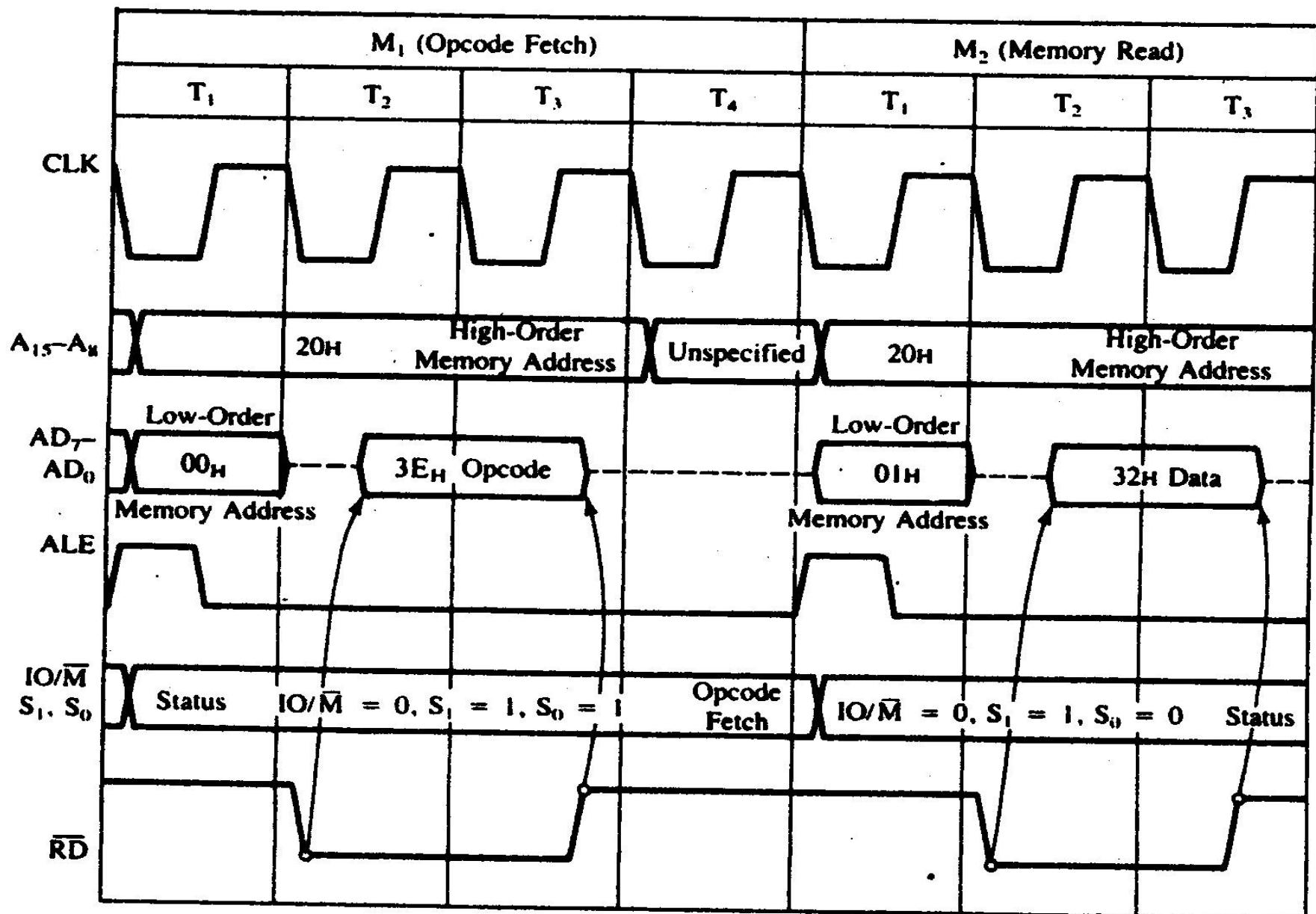


FIGURE 3.10

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

3.3 MEMORY INTERFACING

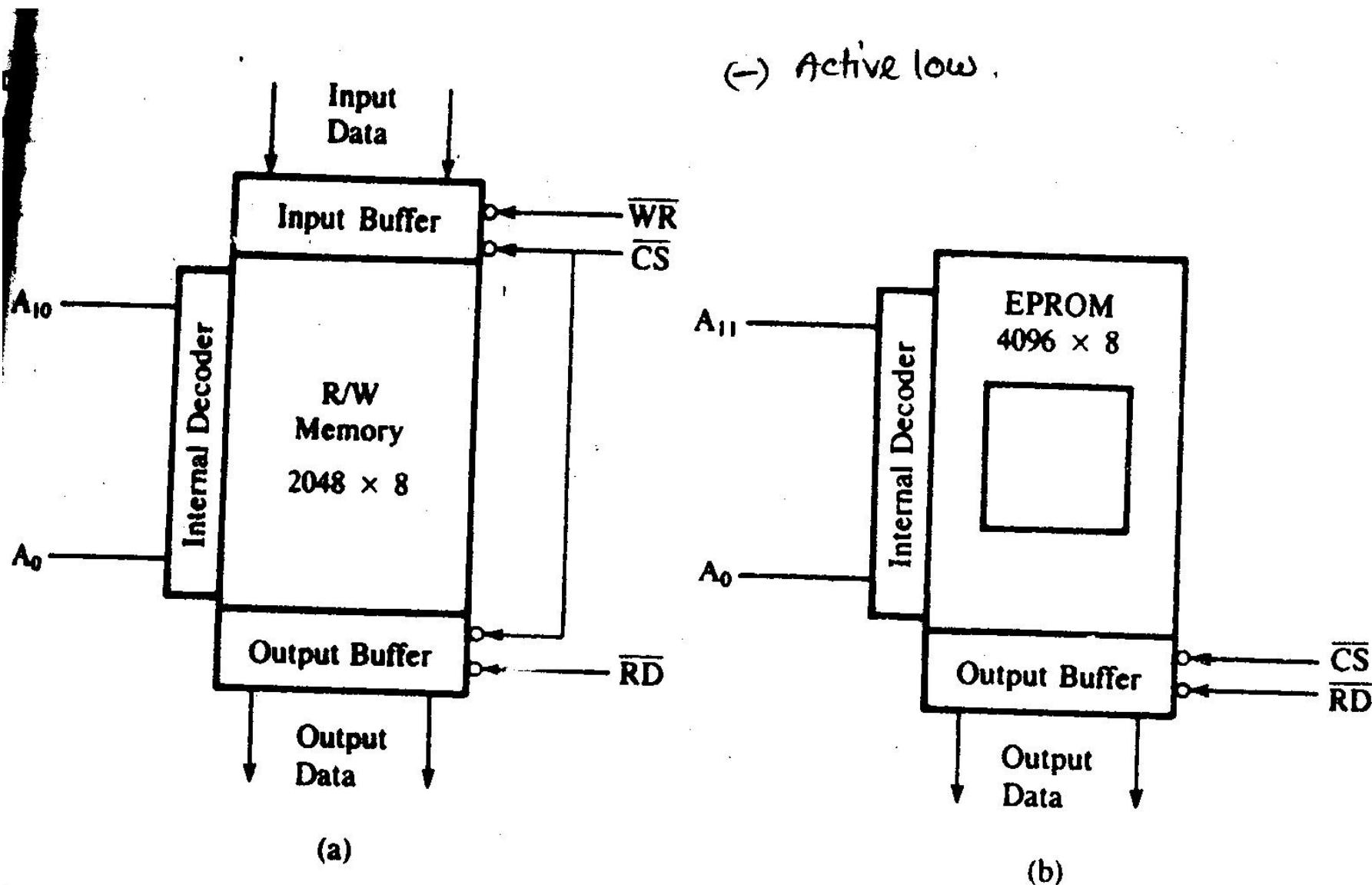
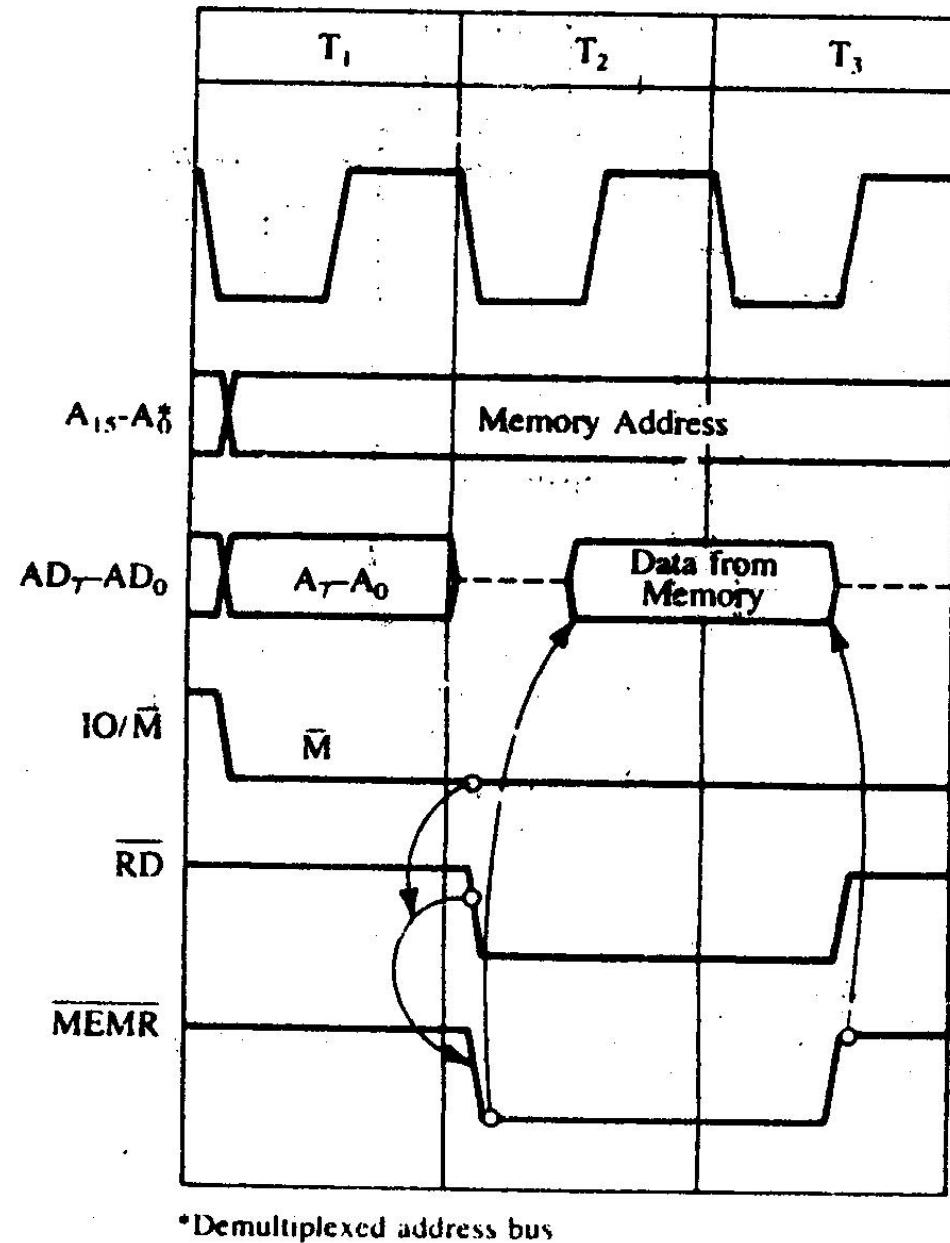


FIGURE 3.11
Typical Memory Chips: R/W Static Memory (a) and EPROM (b)

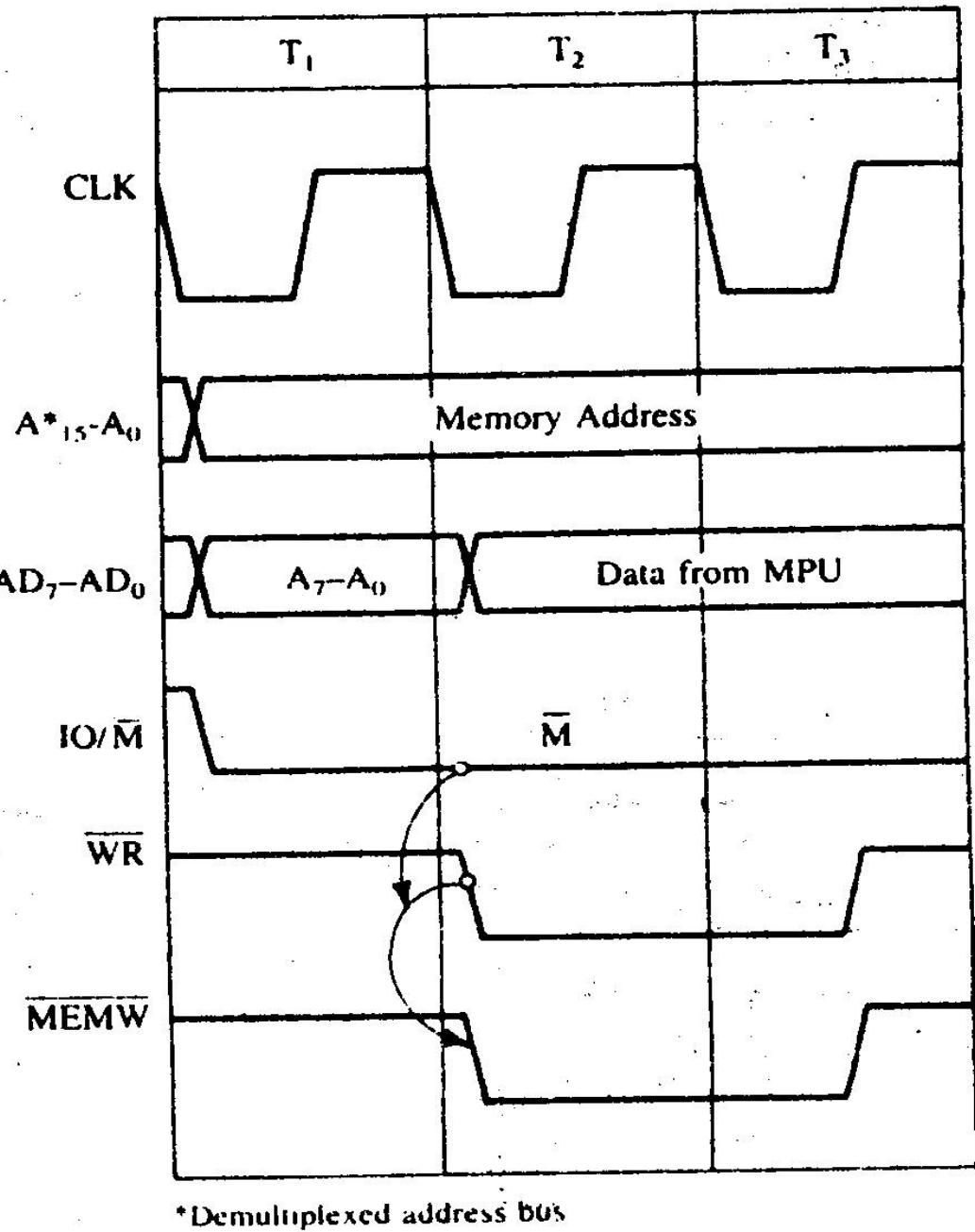
FIGURE 3.12

Timing of the Memory Read Cycle



*Demultiplexed address bus

FIGURE 3.13
Timing of the Memory Write Cycle



Chapter-4

Interfacing I/O Devices

4.1 BASIC INTERFACING CONCEPTS

An I/O device can be interfaced with the 8085 microprocessor either any of the following:

- Peripheral I/O or I/O mapped I/O or Port addressed I/O
- Memory mapped I/O

In peripheral I/O an 8-bit port is used to interface I/O with the processor. Intel 8085 processor uses peripheral I/O for data transfer. But, in memory mapped I/O a 16-bit memory register is used to interface I/O with the processor. Motorola 68000 processor uses memory mapped I/O for data transfer.

Peripheral I/O Instructions

- In peripheral I/O the 8085 microprocessor uses two instructions for data transfer between the processor and the I/O devices. The instructions are IN and OUT.

OUT Instruction:

An OUT instruction is used to give any output to an output device such as monitor, LED etc. The syntax of an OUT instruction is as follows:

OUT 8-bit Port Address

<u>Mnemonic</u>	<u>Machine Code</u>	<u>Memory Address</u>	<u>Memory Contents</u>
OUT 01H	D3 01	2050 2051	11010011 00000001

4.1.2 I/O Execution

OUT INSTRUCTION

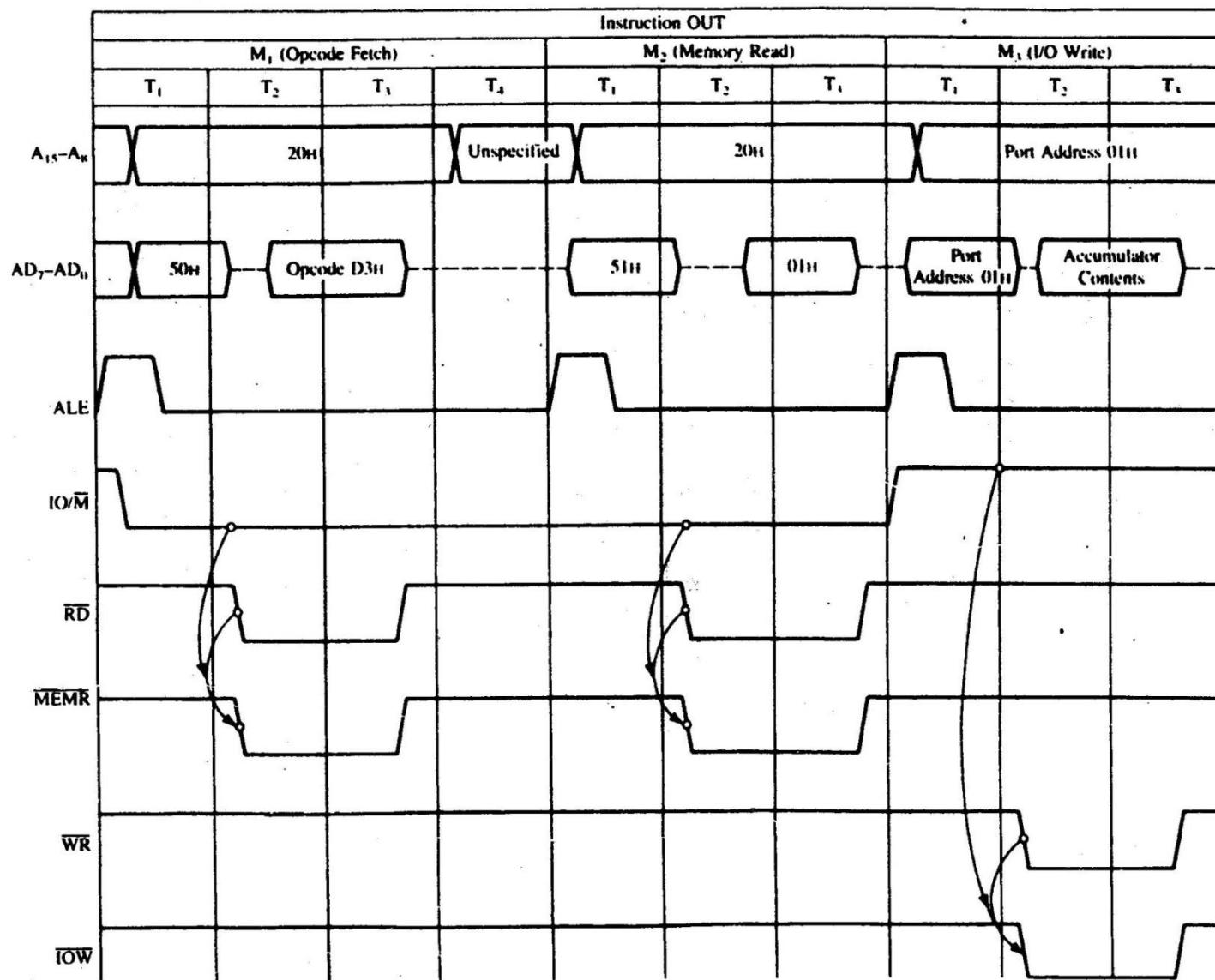
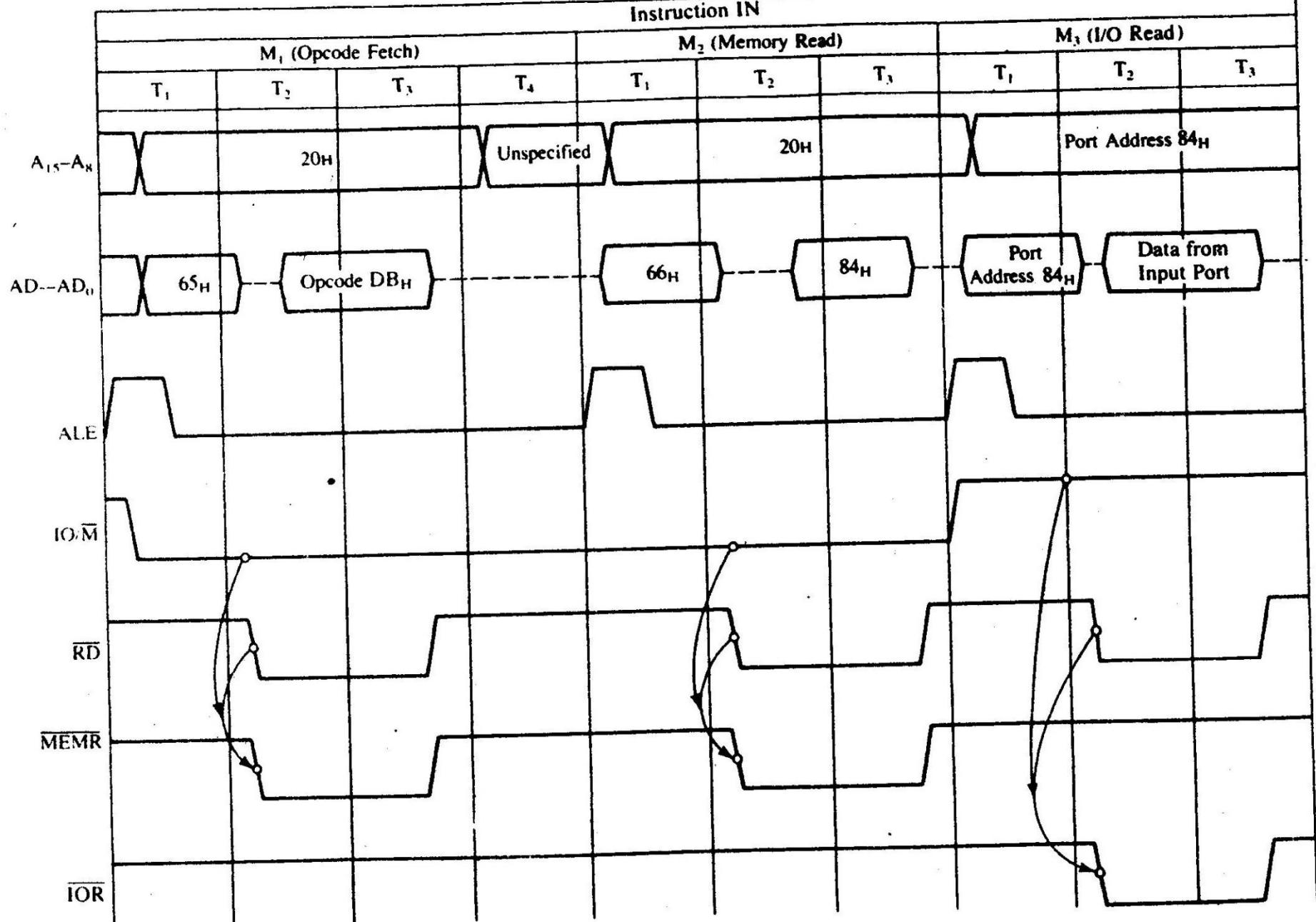


FIGURE 4.1
8085 Timing for Execution of OUT Instruction

IN Instruction: An IN instruction is used to take any input from an input device such as keyboard, switch etc. The syntax of an IN instruction is as follows:

IN 8-bit Port Address

<u>Mnemonic</u>	<u>Machine Code</u>	<u>Memory Address</u>	<u>Memory Contents</u>
IN 84H	DB 84	2065 2066	11011011 10000100



5.1.3 Device Selection and Data Transfer

Two problems regarding the I/O data transfer (M_3 cycle of OUT and IN instruction timing wave form):

- (1) When should we enable the latch (or buffer for IN) to catch the information?
- (2) What should be the address of that latch?

The solution of above problems are as follows:

1. Decode the address bus to generate a unique pulse 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 pulse (I/O select pulse)**.
3. Use the I/O select pulse to activate the interfacing device (I/O port).

Figure 4.3 shows a block diagram and Figure 4.4 shows an example of above solution.

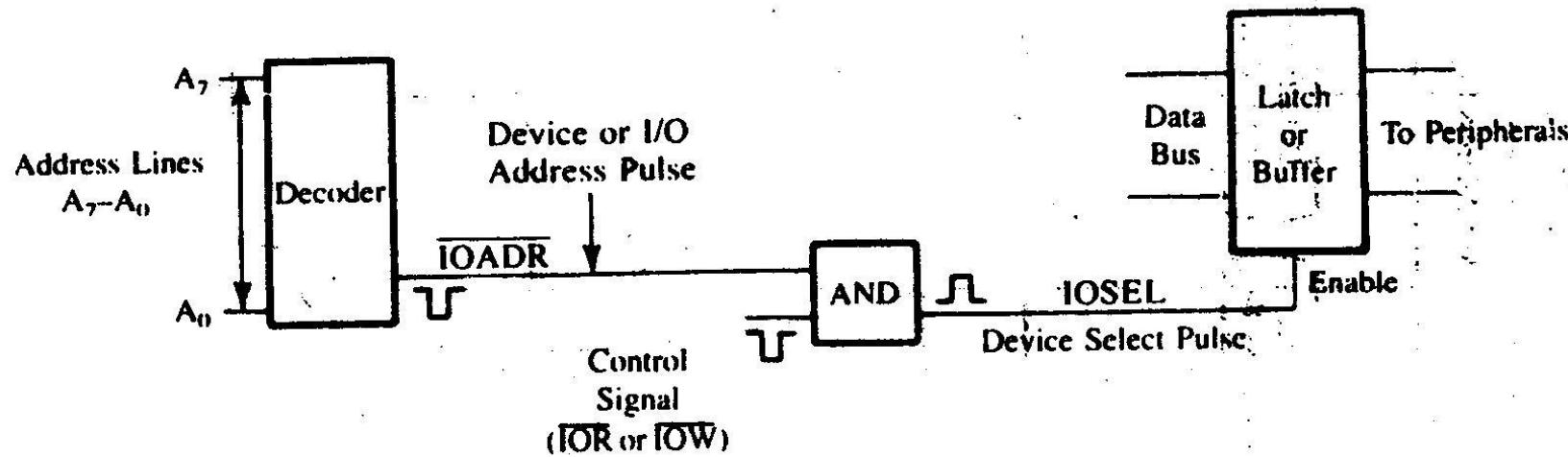


FIGURE 4.3
Block Diagram of I/O Interface

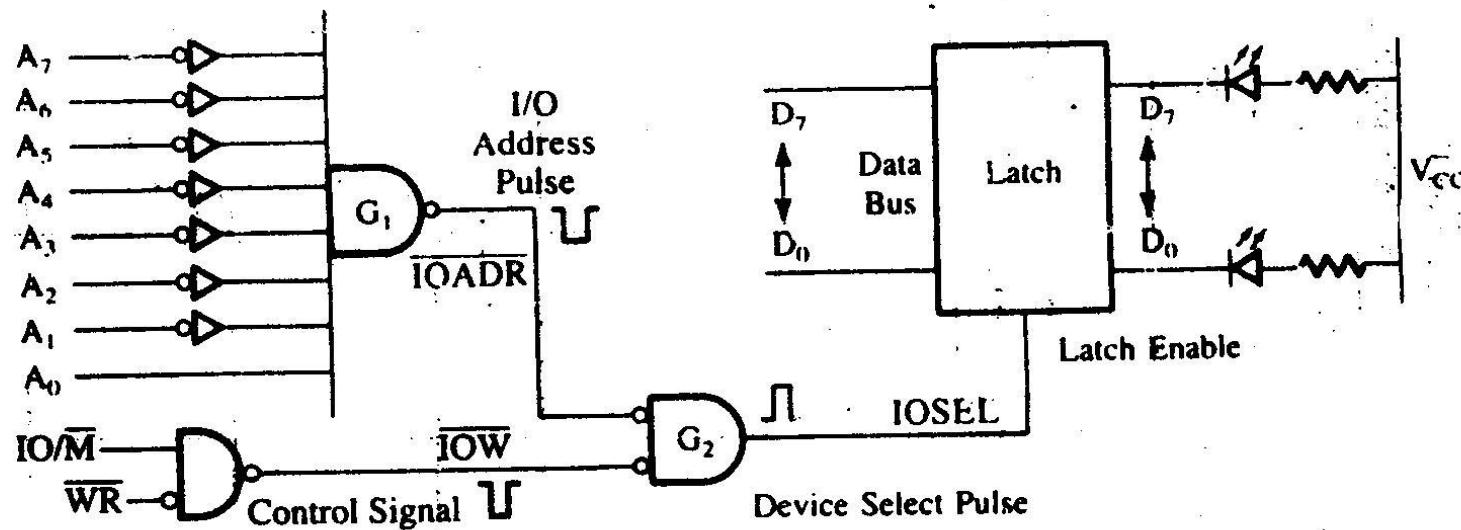


FIGURE 4.4
Decode Logic for LED Output Port
NOTE: To use this circuit with the 8085, the bus AD₇-A₀ must be demultiplexed.

4.14 Absolute Vs. Partial Decoding

- In Figure 4.4, all eight lines are decoded to generate one unique output pulse; the device will be selected only with the address, 01H. This is called **absolute decoding** and good design practice.
- However, to minimize the cost, the output port can be selected by decoding some of the address lines ,as shown in Figure 4.5; this is called **partial decoding**. As a result, the device has multiple addresses.

Figure 4.5 is similar to Figure 4.4 except that the address lines A0 and A1 are not connected, and they are replaced by IO/M and WR signals. Because the address lines A0 and A1 are at don't care logic level, they can be assumed 0 or 1. Thus this output port (latch) can be accessed by the addresses 00H, 01H, 02H, and 03H.

The partial decoding is commonly used technique in small systems. Such multiple addresses will not cause any problem, provided these addresses are not assigned to any other output ports.

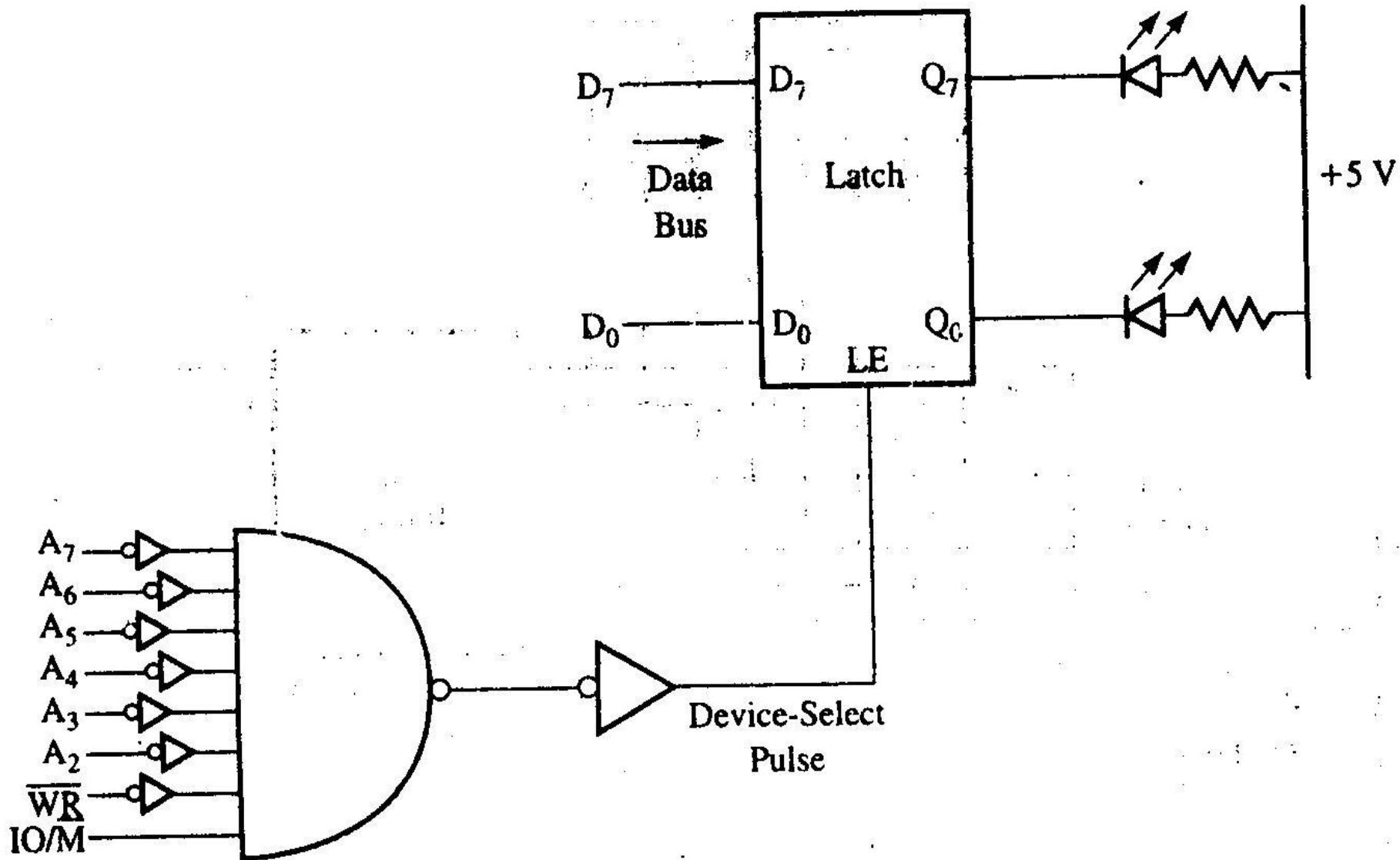


FIGURE 4.5

Partial Decoding: Output Latch with Multiple Addresses

4.15 Input Interfacing

Figure 4.6, shows an example of interfacing an 8-key input port. The circuit for the input port in Figure 4.6 differs from the output port in Figure 4.4 as follows:

1. Control signal \overline{IOR} is used in place of \overline{IOW} .
2. The tri-state buffer is used as an interfacing port in place of the latch.
3. In Figure 4.6, data flow from the keys to the accumulator; on the other hand, in Figure 4.4, data flow from the accumulator to the LEDs.

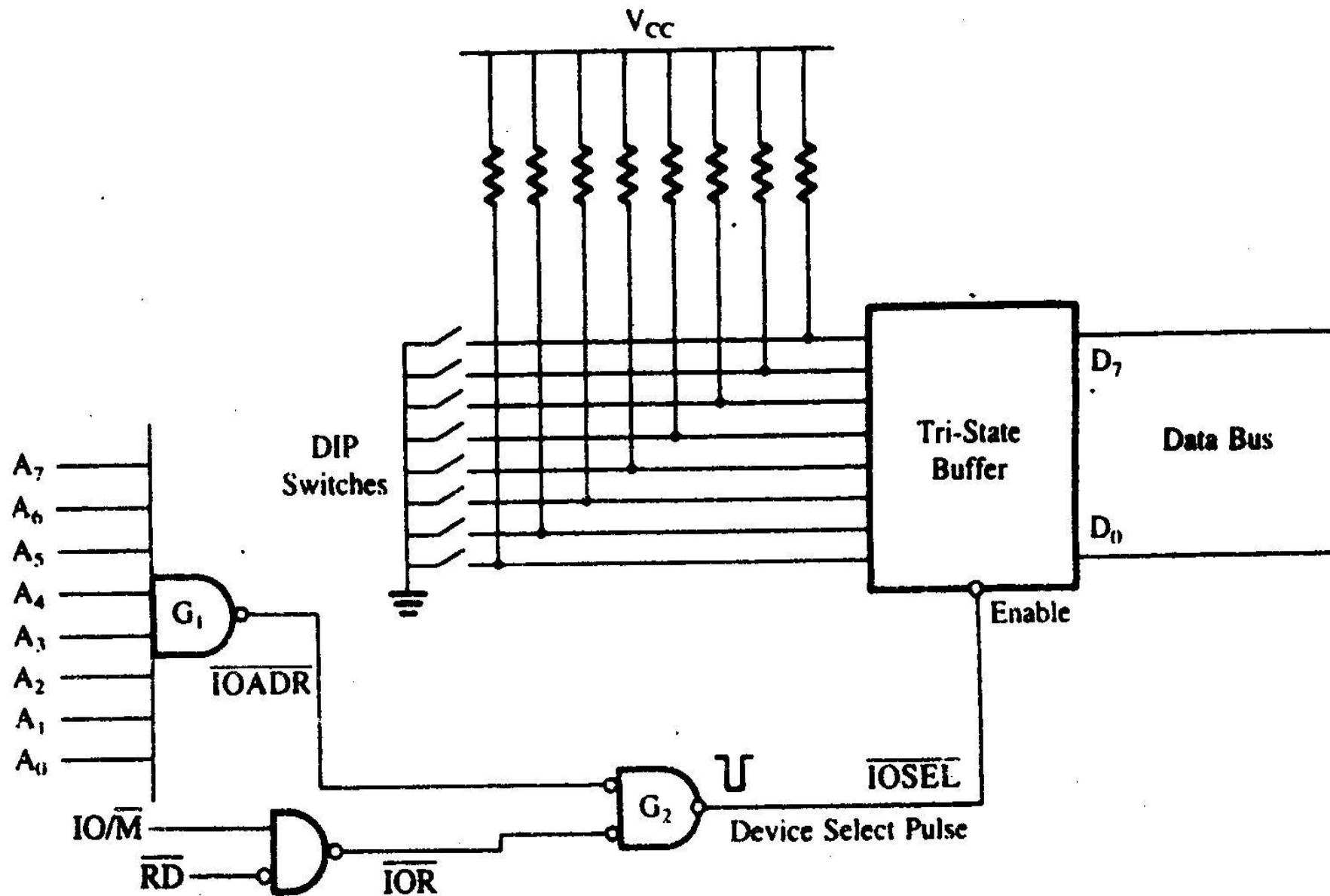


FIGURE 4.6

Decode Logic for a Dip-Switch Input Port

4.16 Interfacing I/Os Using Decoders

Figure 4.7 illustrates how a decoder is used to decode the address of input port and output port. To select or enable the tri-state buffer and latch respectively, following addresses are used to decoded by the decoder (Figure 4.7)

A ₇	A ₆	A ₅	A ₄	A ₃	A ₂	A ₁	A ₀
1	1	1	1	1	0	0	0

= F8H (Latch)

1	1	1	1	1	0	1	0
---	---	---	---	---	---	---	---

= FAH (Buffer)

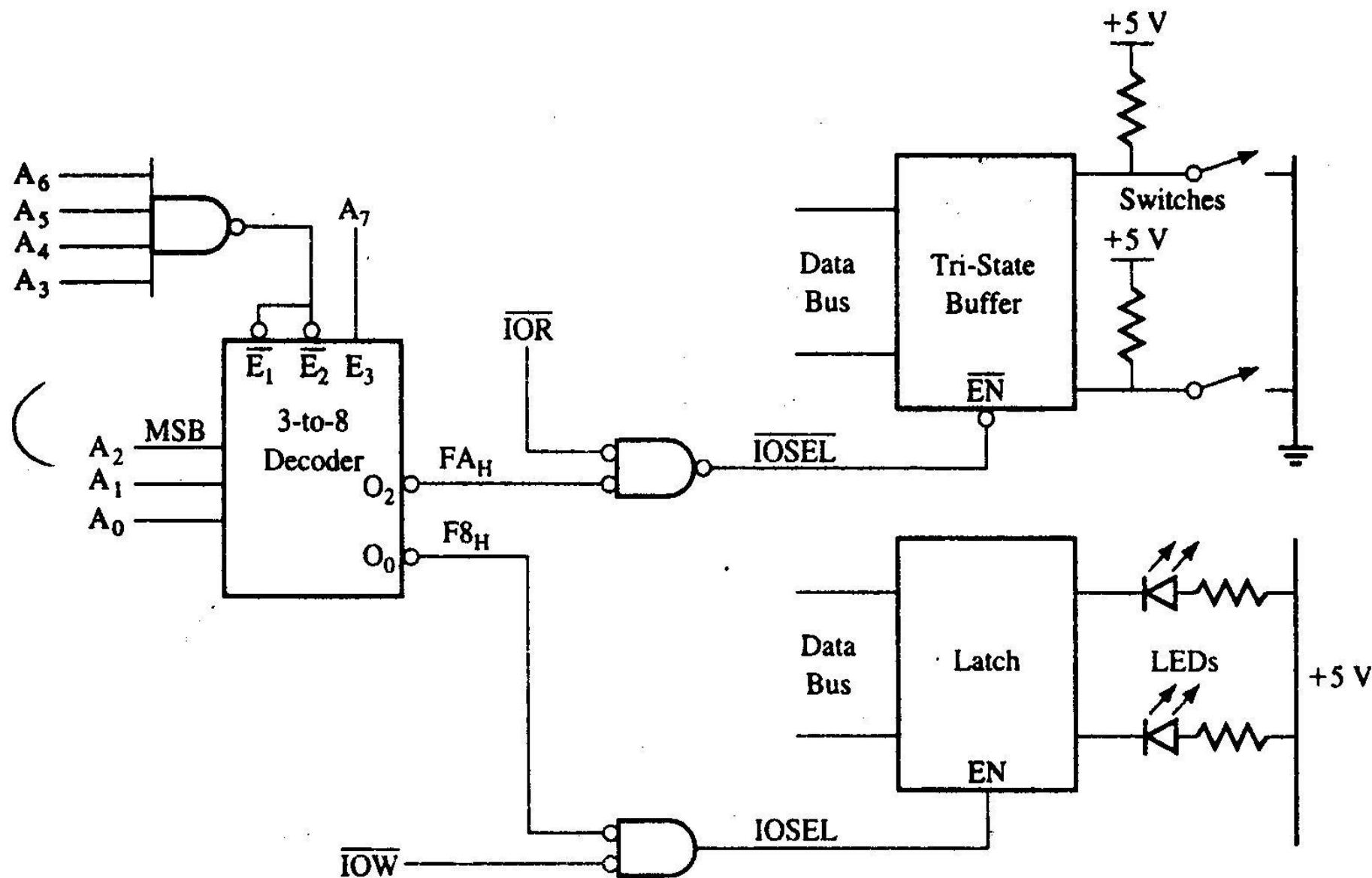


FIGURE 4.7

Address Decoding Using a 3-to-8 Decoder

4.2 INTERFACING OUTPUT DISPLAYS

4.22 Illustration: Seven-Segment LED Display as an Output Device

PROBLEM STATEMENT

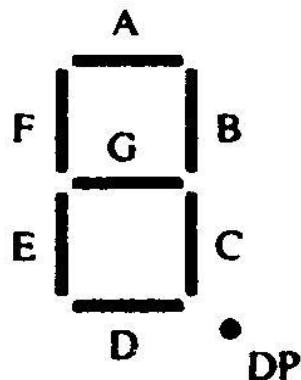
1. Design a seven-segment LED output port with the device address F5H, using a 74LS138 3-to-8 decoder, a 74LS20 4-input NAND gate, a 74LS02 NOR gate, and a common-anode seven-segment LED.
2. Given \overline{WR} and $\overline{IO/M}$ signals from the 8085, generate the \overline{IOW} control signal.
3. Explain the binary codes required to display 0 to F Hex digits at the seven-segment LED.
4. Write instructions to display digit 7 at the port.

HADWARE DESCRIPTION

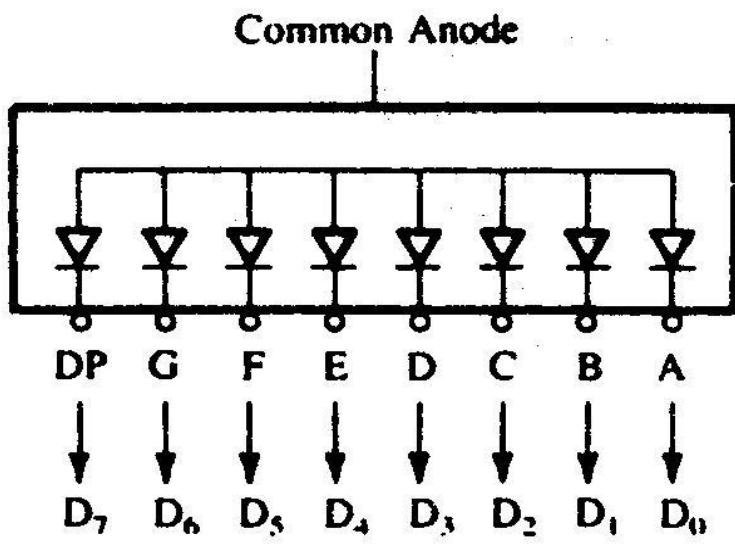
The circuit description is given below:

SEVEN-SEGMENT DISPLAY

A seven-segment LED consists of seven light emitting diode and one segment for the decimal point. The LEDs are physically arranged as shown in Figure 4.9 (a). To display a number, the necessary segments are lit by sending an appropriate signals for current flow through diodes.



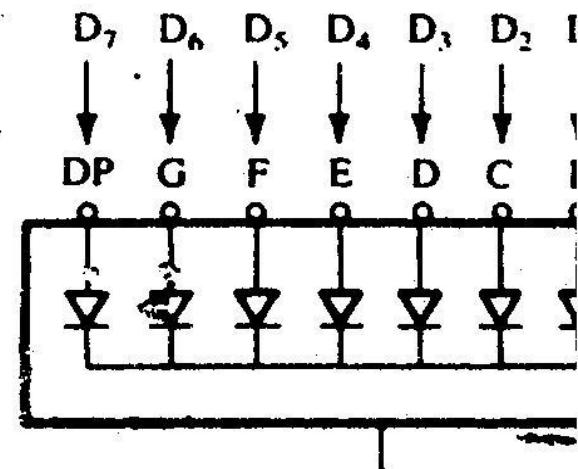
(a)



To Data Lines
Through an Interfacing Device

(b)

From Data Lines
Through an Interfacing Device



Common Cathode

(c)

FIGURE 4.9

Seven-Segment LED: LED Segments (a); Common-Anode LED (b); Common-Cathode

To display digit 7 at the LED in Figure 4.10, the requirements are as follows:

1. It is a common-anode seven-segment LED, and logic 0 is required to turn on a segment.
2. To display digit 7, segments A, B, and C should be turned on.
3. The binary code should be

Data Lines	D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	
Bits	X	1	1	1	1	0	0	0	= 78H
Segments	NC	G	F	E	D	C	B	A	

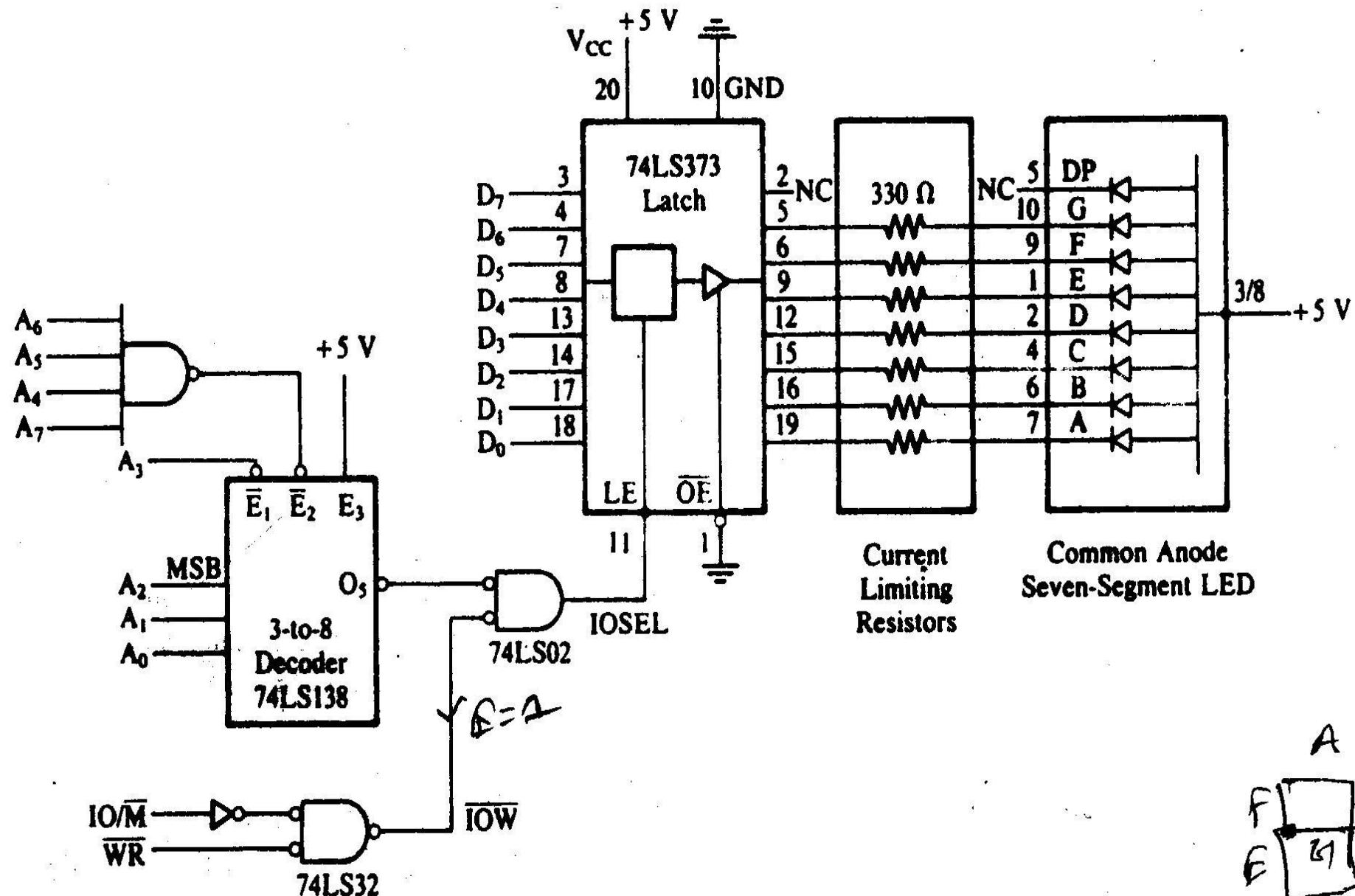
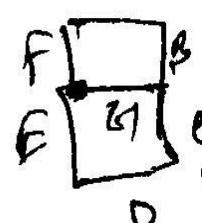


FIGURE 4.10

Interfacing Seven-Segment LED

7



To design an output port with the address F5H, the address lines A₇ – A₀ should have following logic:

A ₇	A ₆	A ₅	A ₄	A ₃	A ₂	A ₁	A ₀
1	1	1	1	0	1	0	1

= F5H

Instructions:

The following instructions are necessary to display 7 at the output port:

MVI A, 78H

OUT F5H

HLT

4.3 INTERFACING INPUT DEVICES

4.31 Illustration: Data Input from DIP Switches

The circuit used for interfacing eight DIP switches, as shown in Figure 4.11. The circuit shows the 74LS138 3-to-8 decoder to decode the low-order bus and tri-state octal buffer (74LS244) to interfaces the switches to the data bus. The port can be accessed with the address 84H.

4.32 Hardware

Figure 4.11 shows the 74LS244 tri-state octal buffer used as an interfacing device. When OE signal goes low, the input data show up on the output lines (connected to the data bus).

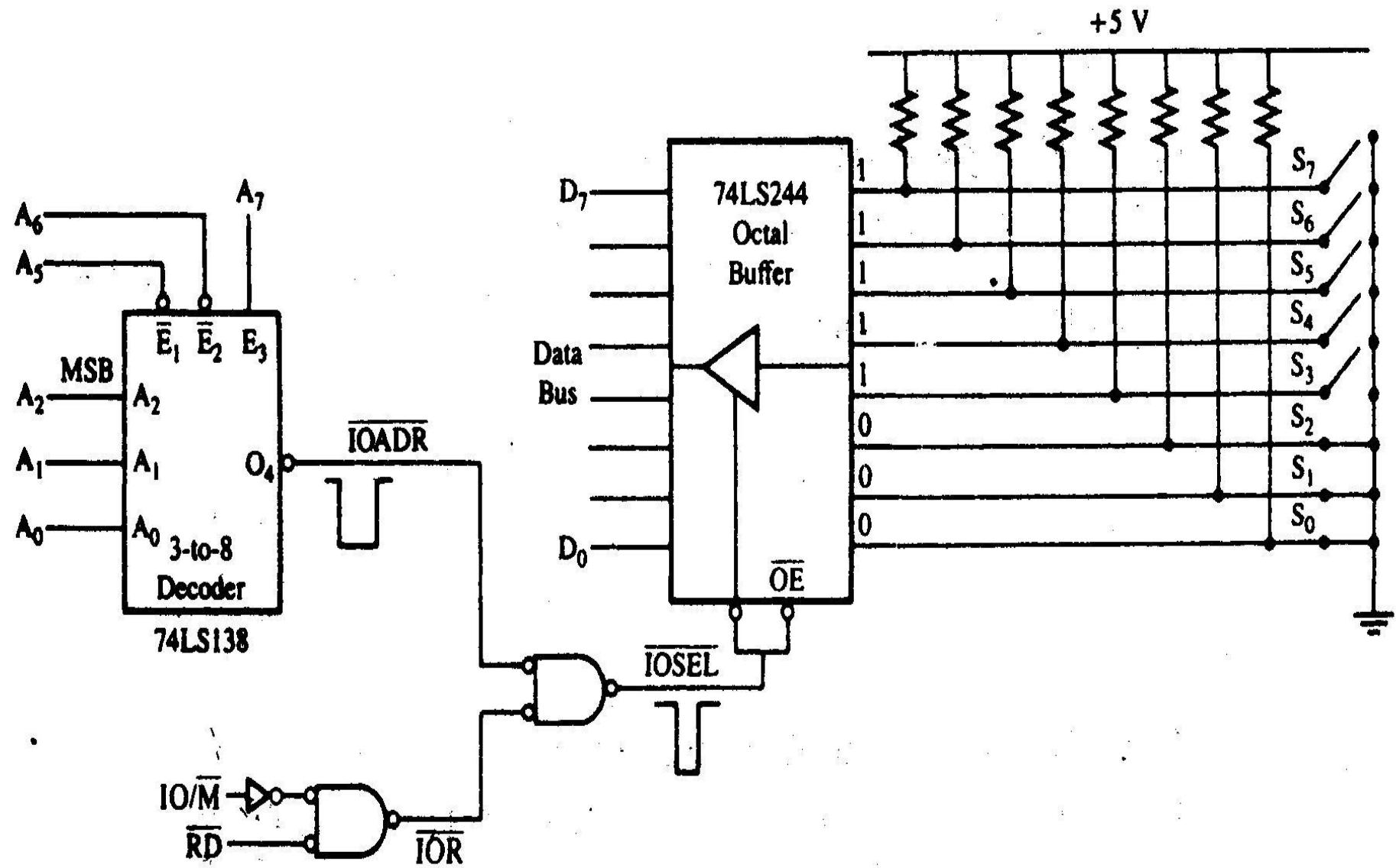


FIGURE 4.11
Interfacing DIP Switches

4.33 Interfacing Circuit

Figure 4.11 shows that the address lines A_4 and A_3 are left in the don't care state. The output line \overline{OE} goes low when the address bus has the following address (assume the don't care lines are at logic 0):

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
1	0	0	0	0	1	0	0

 $= 84H$

4.34 Multiple Port Address

In Figure 4.11, the address lines A_4 and A_3 are not used by the decoding circuit; the logic levels on these lines can 0 or 1. Therefore, this input port can be accessed by four different addresses, as shown below.

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
1	0	0	0	0	1	0	0

 $= 84H$
 $= 8CH$
 $= 94H$
 $= 9CH$

4.35 Instructions to Read Input Port

To read data from the input port shown in Figure 4.11, the instructions are as follows:

IN 84H

HLT

When the instruction IN 84H is executed, during M_3 cycle, the 8085 places the address 84H on the low-order (as well as high-order bus), asserts the RD control signal, and reads the switch positions.

4.4 MEMORY- MAPPED I/O

In memory-mapped I/O, the input and output devices are assigned and identified by 16-bit addresses. To transfer data between the MPU and I/O devices, memory-related instructions (such as LDA, STA etc.) and memory control signals MEMR and MEMW are used.

To understand memory-mapped I/O technique, the following example is considered:

Memory Address	Machine Code	Mnemonics
2050	32	STA 8000H
2051	00	
2052	80	

The STA is a three-byte instructions.

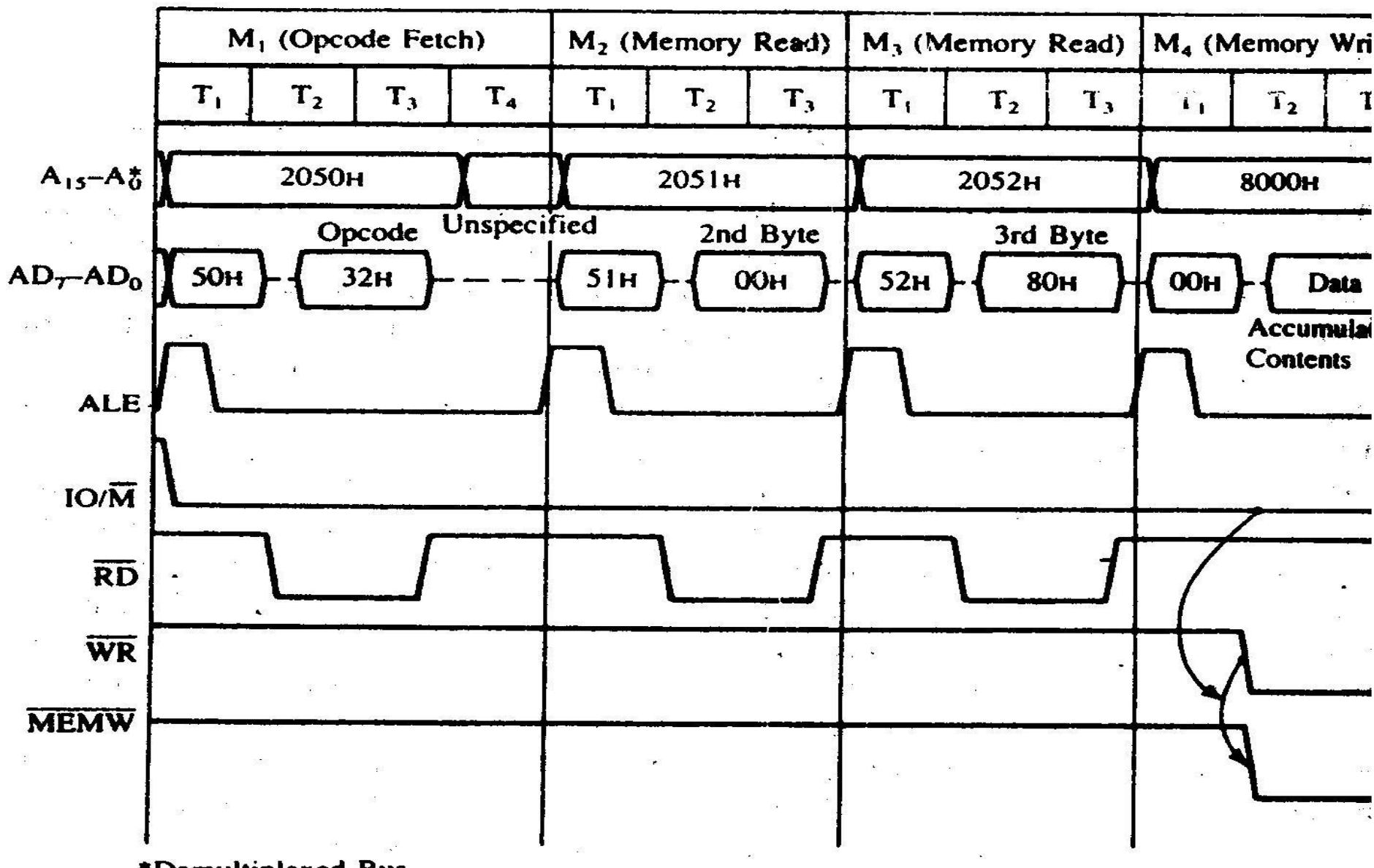
In memory-mapped I/O technique both the memory and I/O devices uses the common domain of addressing. That is an input device or an output device can be connected to the memory location.

4.41 Execution of Memory-Related Data Transfer Instructions

To execute the STA instruction, 8085 microprocessor requires four machine cycles (M_1 to M_4) and thirteen T-states (Figure 4.12). The machine cycle M_4 for the STA instruction is similar to the machine cycle M_3 for the OUT instruction.

Device selection and data transfer in memory-mapped I/O require three steps that are similar to those required in peripheral I/O:

1. Decode the address bus to generate the device address pulse.
2. AND the control signal with the device address pulse to generate the device select pulse.
3. Use the device select pulse to enable the I/O port.



*Demultiplexed Bus

FIGURE 4.12

Timing for Execution of the Instruction: STA 8000H

4.42 Illustration: Safety Control System Using Memory-Mapped I/O Technique

Figure 4.13 shows a schematic of interfacing I/O devices using the memory-mapped I/O technique. The circuit includes one input port with eight DIP switches and one output port to control various processes and gates, which are turned on/off by the microprocessor according to the corresponding switch position.

OUTPUT PORT AND ITS ADDRESS

The various process control devices are connected to the data bus through the latch 74LS373 and solid state relays. When LE is high, the data enter the latch and when LE goes low, data are latched. The latched data are available on the output lines enabled by \overline{OE} . To assert the I/O select pulse, the output port address should be FFF8H, as shown below:

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	= FFF8H
1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	

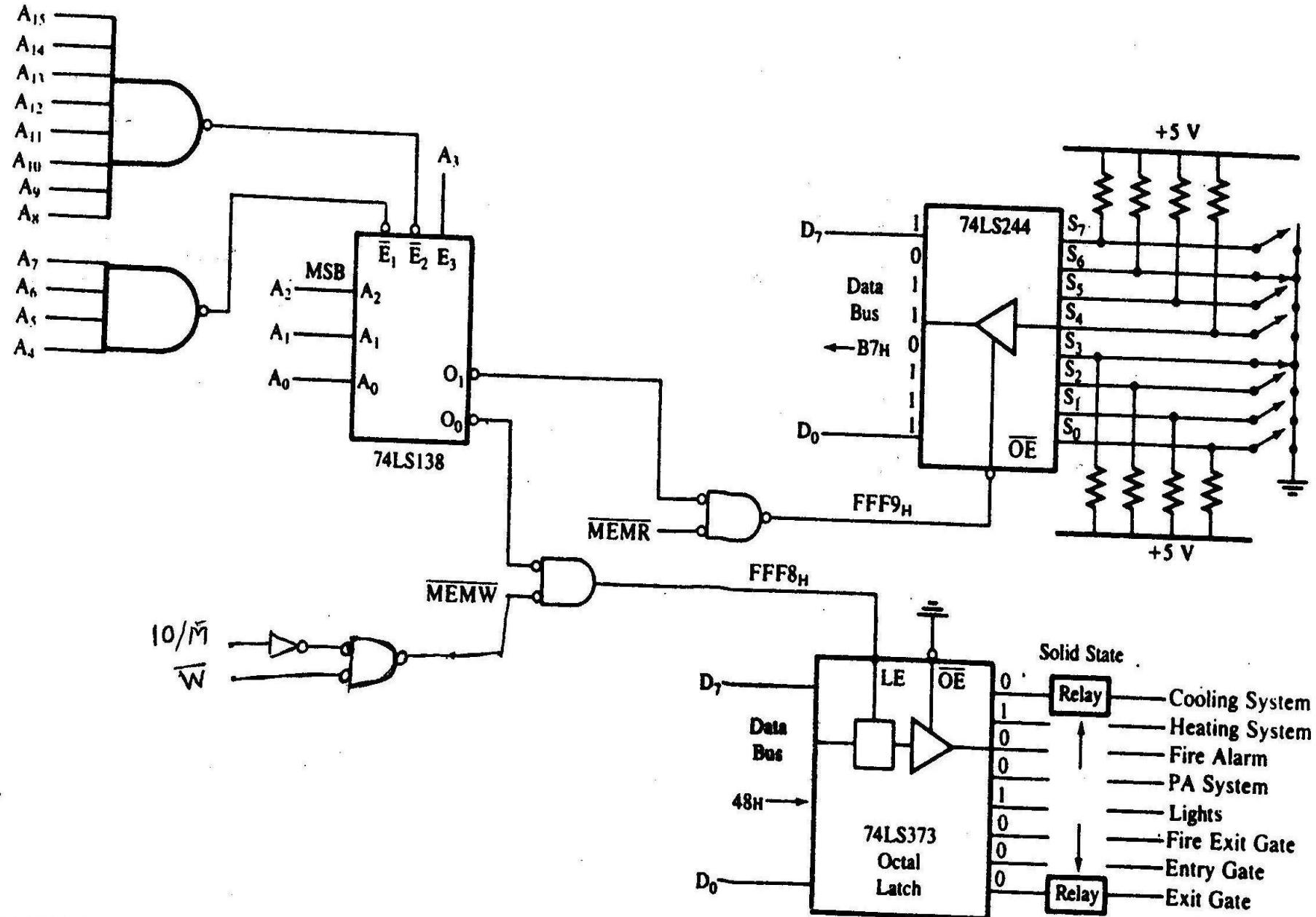


FIGURE 4.13
Memory-Mapped I/O Interfacing

INPUT PORT AND ITS ADDRESS

The DIP switches are interfaced with the 8085 using tri-state buffer 74LS244. The switch positions can be read by enabling the signal \overline{OE} . To read the input port, the port address should be

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
1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1

 $= FFF9H$

Instruction:

To control the processes according to switch positions, the microprocessor should read the bit pattern at the input port and send that bit pattern to the output port. The following instructions can accomplish this task:

READ: LDA FFF9H

CMA

STA FFF8H

JMP READ

Chapter-6

Introduction to 8085 Instructions

Example 6.2:

Write instructions to read eight ON/OFF switches connected to the address 00H, and turn on the devices connected to the output port with the address 01H, as shown in Figure 6.1. (I/O port addresses are given in hexadecimal.)

Solution:

The input has eight switches that are connected to the data bus through the tri-state buffer. Any one of the switches can be connected to +5V (logic 1) or to ground (logic 0), and each switch controls the corresponding device at the output port.

Instructions

```
IN  00H  
OUT 01H  
HLT
```

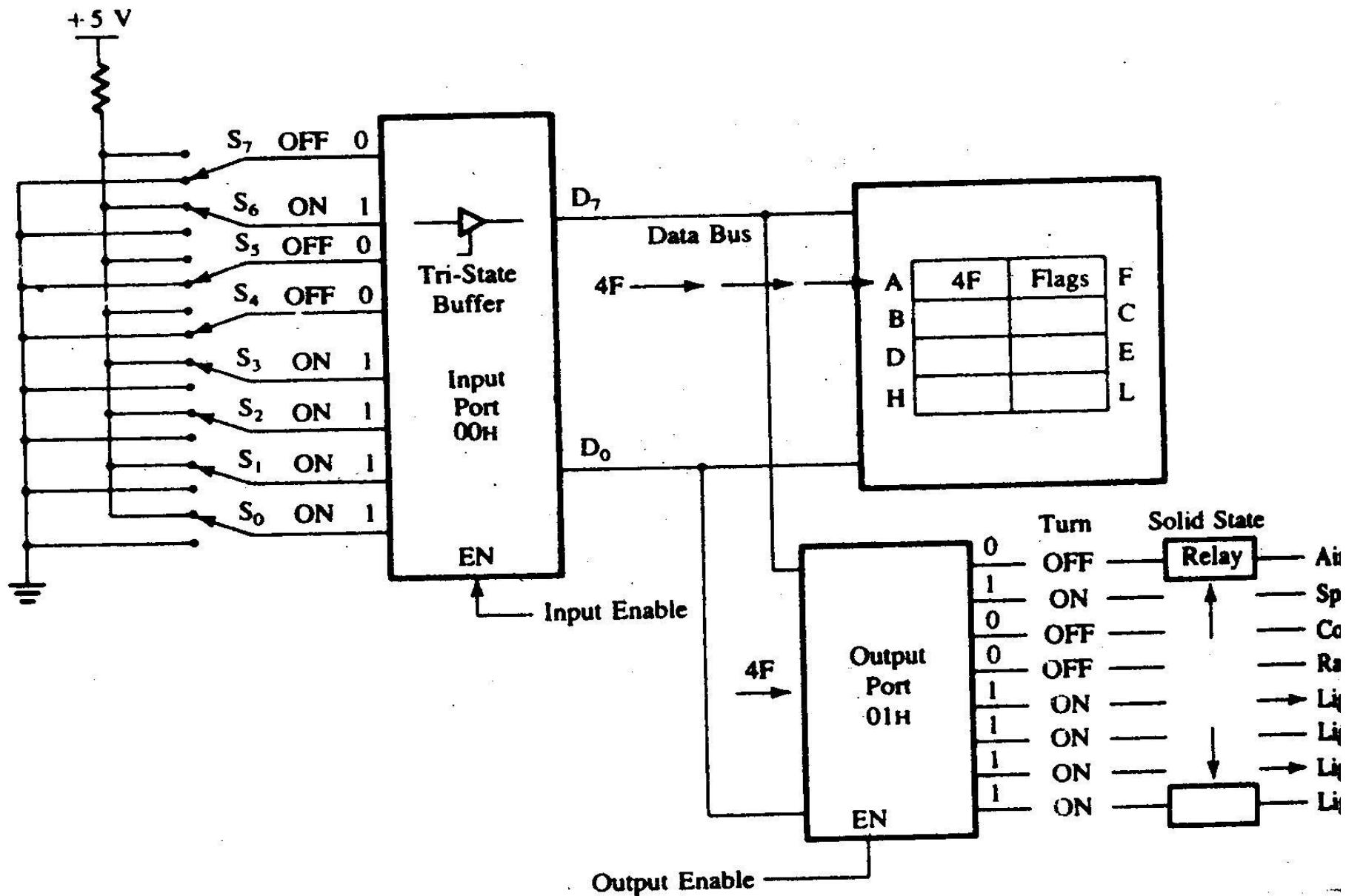


FIGURE 6.1
Reading Data at Input Port and Sending Data to Output Port

Example 6.8:

In Figure 6.8, keep the radio on (D_4) continuously without affecting the functions of other appliances, even if someone turns off the switch S_4 .

Solution

The bit D_4 should be set by Oring the input port with data byte 10H as follows:

$$\begin{array}{rcl} \text{IN } 00\text{H} : (A) & = & D_7 \ D_6 \ D_5 \ D_4 \ D_3 \ D_2 \ D_1 \ D_0 \\ \text{ORI } 10\text{H} : & = & \begin{array}{cccccccc} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{array} \\ & & \hline (A) & = & D_7 \ D_6 \ D_5 & 1 & D_3 \ D_2 \ D_1 \ D_0 \end{array}$$

In Figure 6.8, assume it is winter, and turn off the air conditioner without affecting the other appliances,

$$\begin{array}{rcl} \text{IN } 00\text{H} : (A) & = & D_7 \ D_6 \ D_5 \ D_4 \ D_3 \ D_2 \ D_1 \ D_0 \\ \text{ANI } 7\text{FH} : & = & \begin{array}{cccccccc} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array} \\ & & \hline (A) & = & 0 & D_6 \ D_5 \ D_4 \ D_3 \ D_2 \ D_1 \ D_0 \end{array}$$

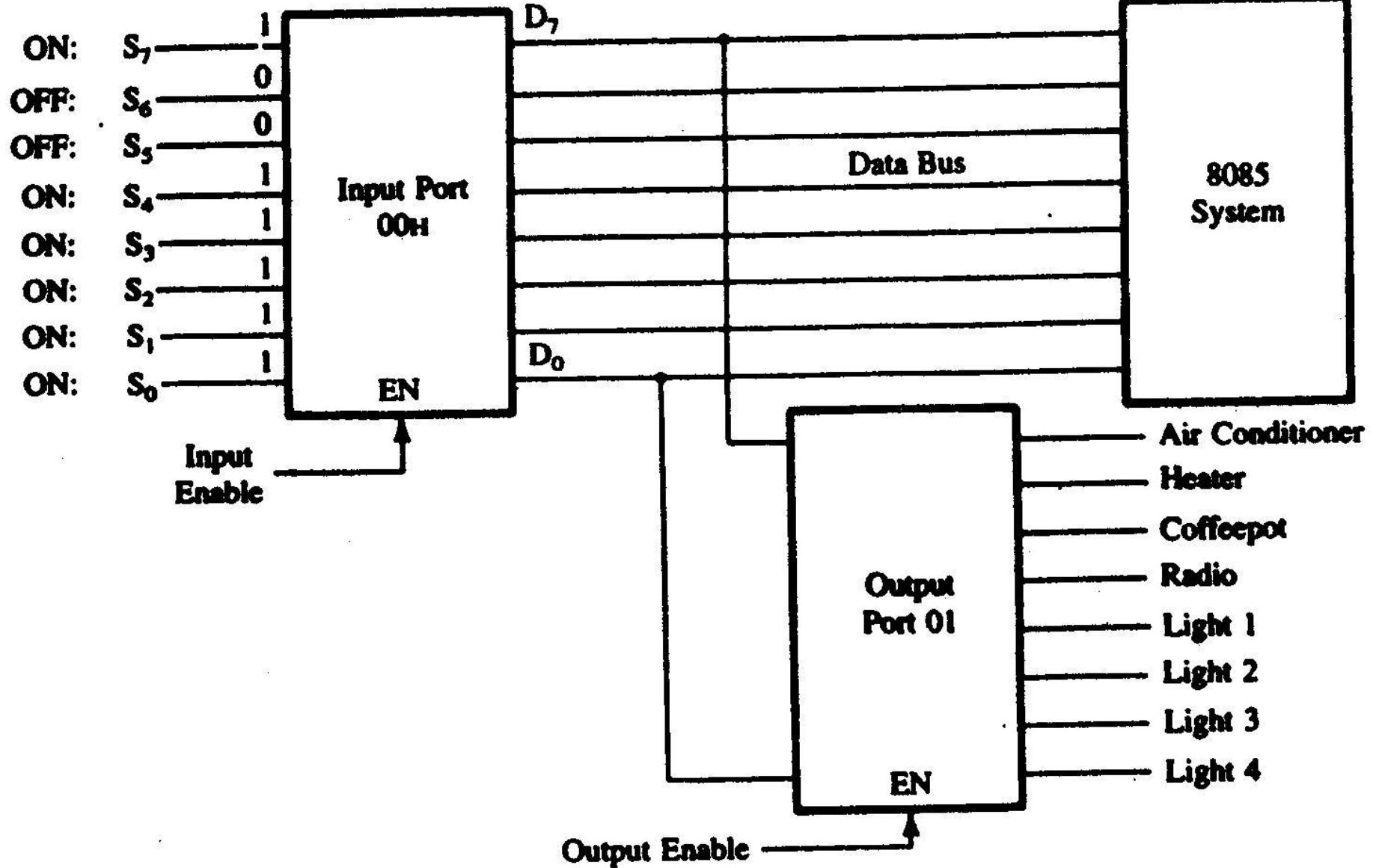


FIGURE 6.8
 Input Port to Control Appliances

6.35 Illustrative Program: ORing Data from Two Input Ports

PROBLEM STATEMENT

An additional input port with eight switches and the address 01H (Figure 6.9) is connected to the microcomputer shown in Figure 6.8 to control the same appliances and lights from the bedroom as well as from the kitchen. Write instructions turn on the devices from any of the input ports.

PROBLEM ANALYSIS

To turn on the appliances from any of the input ports, the microprocessor needs to read the switches at both ports and logically OR the switch positions.

PROGRAM

```
IN    00H  
MOV  B,A  
IN    01H  
ORA  B  
OUT  01H  
HLT
```

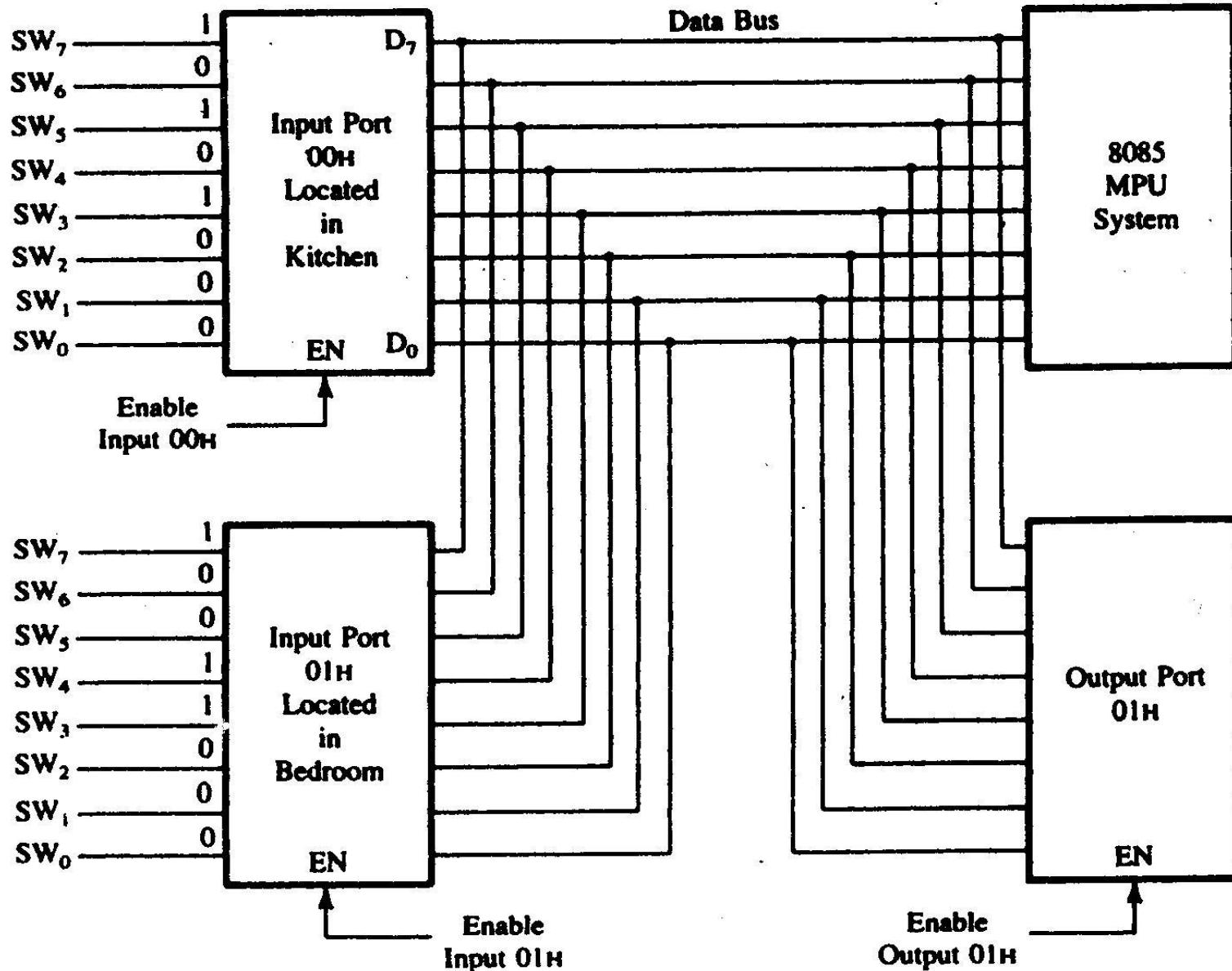


FIGURE 6.9
Two Input Ports to Control Output Devices

6.52 Illustrative Program: Microprocessor-Controlled Manufacturing Process

PROBLEM STATEMENT

A microcomputer is designed to monitor various processes (conveyer belts) on the floor of a manufacturing plant, presented schematically in Figure 6.12.

Write a program to

1. Turn on the five conveyer belts according to the ON/OFF positions of switches $S_4 - S_0$ at port F1H.
2. Turn off the conveyer belts and turn on the emergency alarm only when both switches – S_7 from port F1H an S_7' from port F2H – are triggered.
3. Monitor the switches continuously.

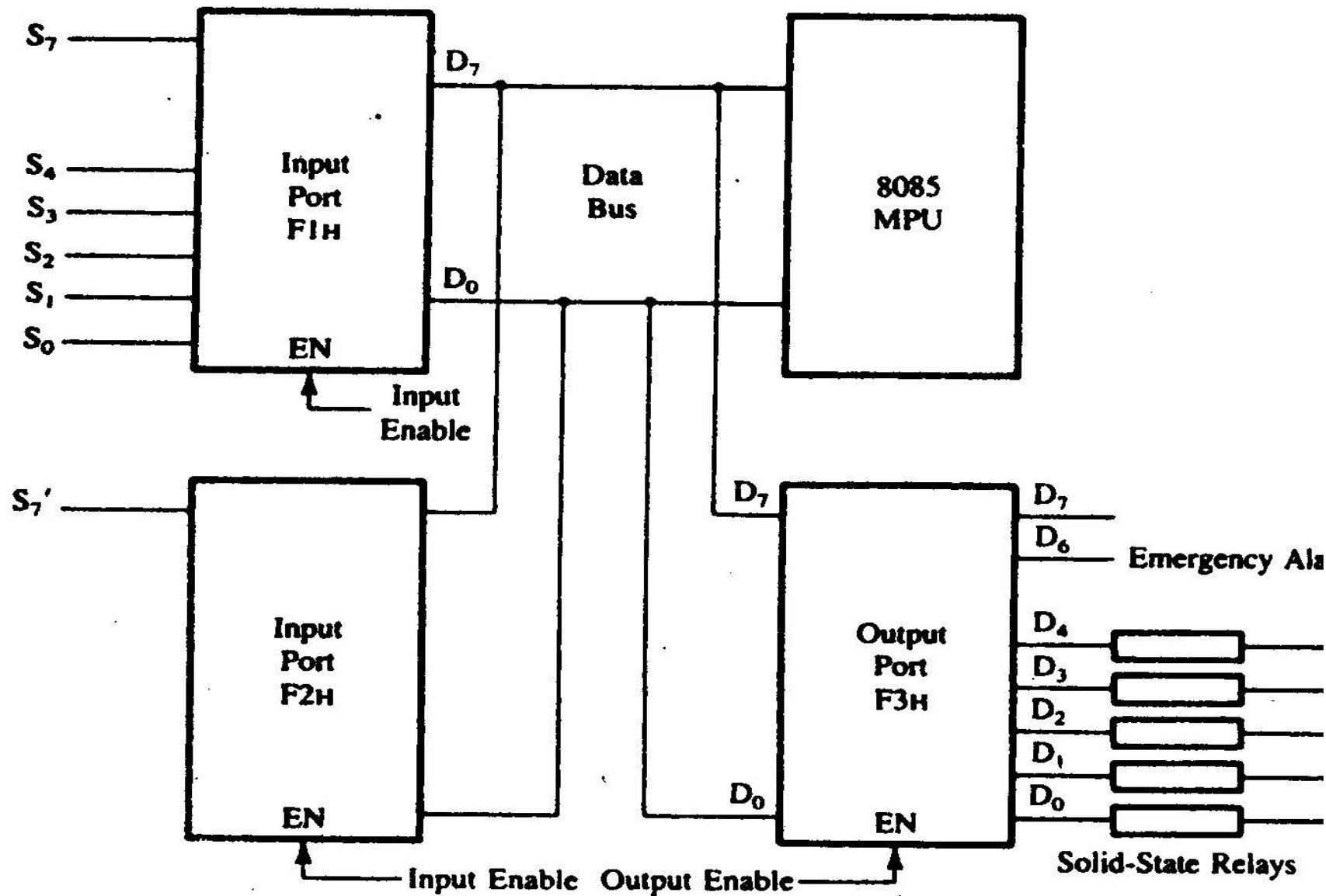


FIGURE 6.12
Input/Output Ports to Control Manufacturing Processes

PROBLEM ANALYSIS

To perform the tasks specified in the problem, the microprocessor needs to

1. read the switch positions.
2. check whether switches S_7 and S_7' from the ports F1H and F2H are on.
3. turn on the emergency signal if both switches are on, and turn off all the conveyer belts.
4. turn on the conveyer belts according to the switch positions S_0 through S_7 at input port F1H if both switches S_7 and S_7' are not on simultaneously.
5. continue checking the switch positions.

FLOWCHART AND PROGRAM

The five steps listed above can be translated into flowchart and an assembly language program as shown in Figure 6.13.

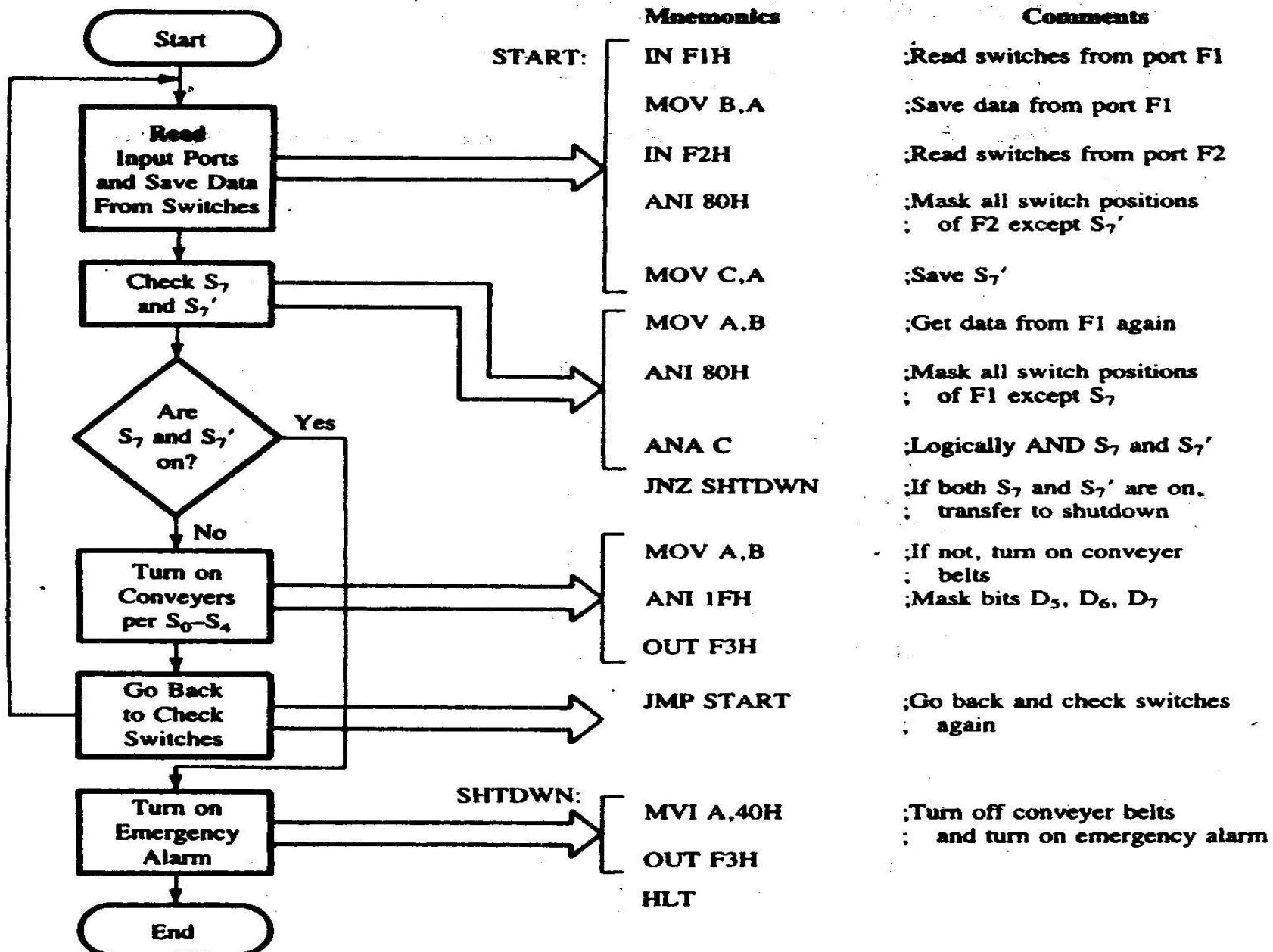


FIGURE 6.13
Flowchart and Program for Controlling Manufacturing Processes

Chapter-8

Counter and Time Delays

8.1 COUNTER AND TIME DELAYS

Counters are used primarily to keep track of events; time delays are important in setting up reasonably accurate timing between two events.

COUNTER

A counter is designed simply by loading an appropriate number into one of the registers and using INR (for up-counter) or DCR (for down-counter) instructions. A loop is established to update the count.

TIME DELAY

The procedure used to design a specific delay is similar to that used to set up a counter. A register is loaded with a number, depending on the time delay required, and then the register is decremented until it reaches zero.

8.11 Time Delay Using One Register

<u>Label</u>	<u>Opcode</u>	<u>Operand</u>	<u>T-states</u>
	MVI	C,FFH	
	7		
LOOP:	DCR	C	4
	JNZ	LOOP	10/7=3

An 8085-based microcomputer with 2 MHz clock frequency will execute the instruction MVI in 3.5 μ s as follows:

Clock frequency of the system $f = 2 \text{ MHz} = 2 * 10^6 \text{ Hz}$

Clock period $T = 1/f = 1/2 * 10^6 = 0.5 \mu\text{s} = 0.5 * 10^{-6} \text{ s}$

$$\begin{aligned}\text{Time to execute MVI} &= 7 \text{ T-states} \times 0.5 \\ &= 3.5 \mu\text{s}\end{aligned}$$

The time delay in the loop T_L with 2 MHz clock frequency is calculated as

$$T_L = (T \times \text{Loop T-states} \times N_{10})$$

Where T_L = Time delay in the loop

T = System clock period

N_{10} = Decimal equivalent of the count loaded in the delay register

$$\begin{aligned}T_L &= (0.5 \times 10^{-6} \times 14 \times 255) \\&= 1785 \mu\text{s} \\&\approx 1.8 \text{ ms}\end{aligned}$$

The T-states for JNZ instruction are 10/7. So the adjusted loop delay is

$$\begin{aligned}T_{LA} &= T_L - (3 \text{ T-states} \times \text{Clock period}) \\&= 1785 \mu\text{s} - 1.5 \mu\text{s} = 1783.5 \mu\text{s}\end{aligned}$$

Therefore, the total delay is

Total Delay = Time to execute instructions outside loop + Time to execute loop instructions

$$\begin{aligned}T_D &= T_O + T_{LA} \\&= (7 \times 0.5 \mu\text{s}) + 1783.5 \mu\text{s} = 1787 \mu\text{s} \\&\approx 1.8 \text{ ms}\end{aligned}$$

The difference between the loop delay T_L and these calculations is only 2 μs and can be ignored in most cases.

8.12 Time Delay Using a Register Pair

Here the counter value can be a 16-bit number and maximum of FFFFH.

<u>Label</u>	<u>Opcode</u>	<u>Operand</u>	<u>T-states</u>
LOOP:	LXI	B,2384H	10
	DCX	B	6
	MOV	A,C	4
	ORA	B	4
	JNZ	LOOP	10/7

TIME DELAY

The loop includes 24 clock periods for execution. The decimal equivalent of counter value is

$$2384H = 9092_{10}$$

If the clock period of the system = 0.5 μ s, the delay in the loop T_L is

$$\begin{aligned}T_L &= (T \times \text{Loop T-states} \times N_{10}) \\&= (0.5 \times 24 \times 9092_{10}) \\&\approx 109 \text{ ms (without adjusting for the last cycle)}\end{aligned}$$

Total Delay $T_D = 109 \text{ ms} + T_O$
 $\approx 109 \text{ ms}$ (the instruction LXI adds only 5 μs)

8.13 Time Delay Using a Loop within a Loop Technique

<u>Label</u>	<u>Opcode</u>	<u>Operand</u>	<u>T-states</u>
LOOP2:	MVI	B,38H	7
	MVI	C,FFH	7
LOOP1:	DCR	C	4
	JNZ	LOOP1	10/7
	DCR	B	4
	JNZ	LOOP2	10/7

DELAY CALCULATIONS

The delay in LOOP1 is $T_{L1} = 1783.5 \mu\text{s}$. The counter value of LOOP2 is 56_{10} (38H). Delay for the LOOP2 is calculated as follows:

$$\begin{aligned}
 T_{L2} &= 56 (T_{L1} + 21 \text{ T-states} \times 0.5 \mu\text{s}) \\
 &= 56 (1783.5 \mu\text{s} + 10.5 \mu\text{s}) \\
 &= 100.46 \text{ ms}
 \end{aligned}$$

8.3 ILLUSTRATIVE PROGRAM: ZERO-TO-NINE (MODULO TEN) COUNTER

PROBLEM STATEMENT

Write a program to count from 0 to 9 with a one-second delay between each count. At the count of 9, the counter should reset itself to 0 and repeat the sequence continuously. Assume the clock frequency of the microprocessor is 1 MHz.

Delay Calculations

Loop Delay $T_L = 24 \text{ T-states} \times T \times \text{Count}$

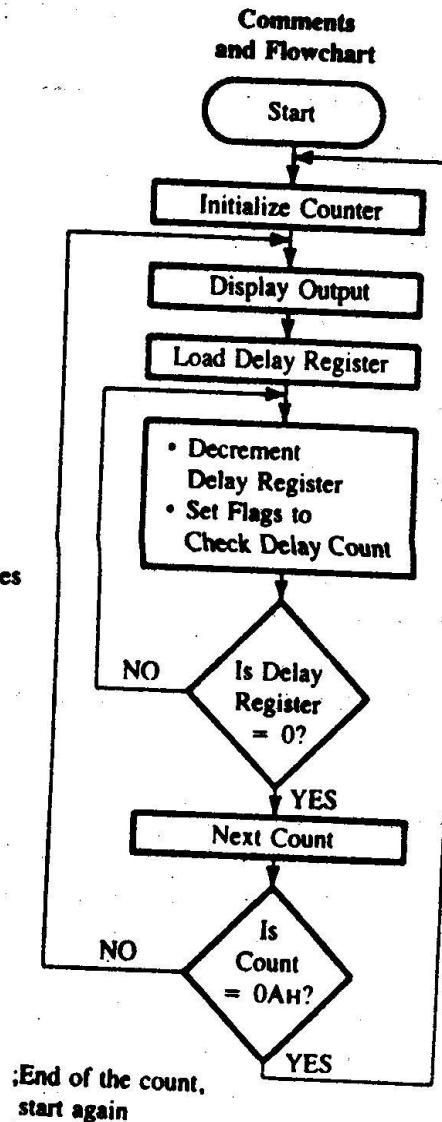
$$1 \text{ second} = 24 \times 1.0 \times 10^{-6} \times \text{Count}$$

$$\text{Count} = 1/(24 \times 10^{-6})$$

$$= 41666_{10}$$

$$= A2C2H$$

Memory Address	Hex Code	Label	Mnemonics	T	
HI-LO XX00					
01 06		START:	MVI B,00H		
02 00					
03 D3		DSPLAY:	OUT PORT#	10	
04 PORT#			LXI H,16-Bit	10	
05 21					T_0
06 LO*					
07 HI					
08 2B		LOOP:	DCX H	6	
09 7D			MOV A,L	4	
0A B4			ORA H	4	
					T_L : 24 T-states
0B C2			JNZ LOOP	10/7	
0C 08					
0D XX†					
0E 04			INR B	4	
0F 78			MOV A,B	4	
10 FE			CPI 0AH	7	
11 0A					T_0
12 C2			JNZ DSPLAY	10/7	
13 03					
14 XX†					
15 CA			JZ START		
16 00					
17 XX					



*Enter 16-bit delay count in place of LO and HI, appropriate to the clock period in your system.
†Enter high-order address (page number) of your R/W memory.

FIGURE 8.7

Program and Flowchart for a Zero-to-Nine Counter

Chapter-9

Stack and Subroutines

9.1 STACK

The stack in an 8085 microcomputer system can be described as a set of memory locations in the R/W memory, specified by programmer in main memory. The beginning of the stack is defined in program by using the instruction LXI SP, 16-bit memory address of a stack.

INSTRUCTIONS

<u>Opcode</u>	<u>Operand</u>
LXI	SP, 16-bit memory address of a stack
PUSH	Rp
PUSH	B
PUSH	D
PUSH	H
PUSH	PSW

<u>Opcode</u>	<u>Operand</u>
---------------	----------------

POP	Rp
POP	B
POP	D
POP	H
POP	PSW

9.2 SUBROUTNE

A subroutine is a group of instructions written separately from main program to perform a function that occurs repeatedly in the main program. A subroutine is like a procedure or function of a high-level language and it is called using the instruction CALL 16-bit memory address of a subroutine.

INSTRUCTIONS

<u>Opcode</u>	<u>Operand</u>
---------------	----------------

CALL	16-bit memory address of a subroutine
RET	

9.21 Illustrative Program: Traffic Signal Controller

PROBLEM STATEMENT

Write a program to provide the given on/off time to three traffic lights (Green, Yellow, and Red) and two pedestrian signs (WALK and DON'T WALK). The signal lights and signs are turned on/off the data bits of an output port as shown below:

<u>Lights</u>	<u>Data Bits</u>	<u>On Time</u>
1. Green	D0	15 seconds
2. Yellow	D2	5 seconds
3. Red	D4	20 seconds
4. WALK	D6	15 seconds
5. DON'T WALK	D7	25 seconds

PROBLEM ANALYSIS

<u>Time</u>	<u>DON'T WALK</u>	<u>WALK</u>	<u>Red</u>	<u>Yellow</u>			<u>Green</u>	<u>Hex</u>
0 ↓ (15)	D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
15 ↓ (5)	0	1	0	0	0	0	0	1 = 41H
20 ↓ (20)	1	0	0	0	0	1	0	0 = 84H
40	1	0	0	1	0	0	0	0 = 90H

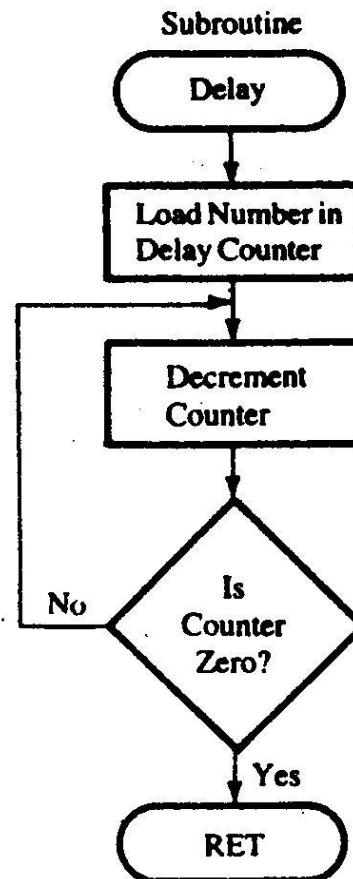
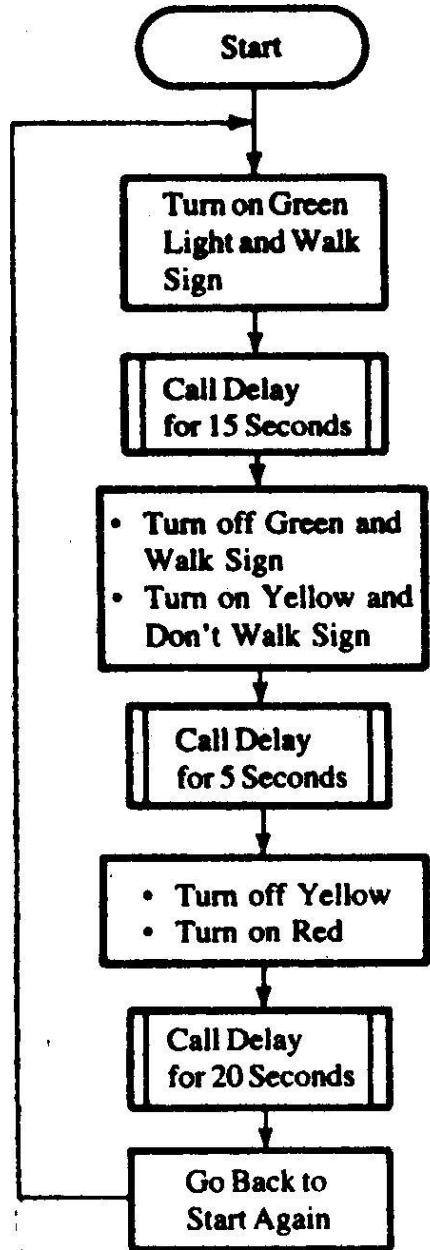


FIGURE 9.13

Flowchart for Traffic Signal Control

PROGRAM

LXI SP, XX99H

START: MVI A, 41H
OUT PORT#

MVI B, 0FH
CALL DELAY

MVI A, 84H
OUT PORT#

MVI B, 05H
CALL DELAY

MVI A, 90H
OUT PORT#

MVI B, 14H
CALL DELAY

JMP STATR

DELAY: PUSH D
 PUSH PSW

SECOND: LXI D, COUNT
 CALL DELAY

LOOP: DCX D
 MOV A,D
 ORA E
 JNZ LOOP

 DCR B
 JNZ SECOND

 POP PSW
 POP D
 RET

Chapter-12

Interrupts

12.1 THE 8085 INTERRUPTS

The 8085 interrupt process is controlled by the Interrupt Enable flip-flop, which is internal to the processor and can be set or reset by using software instructions. If the flip-flop is enabled and the input to the interrupt signal INTR (pin 10) goes high, the microprocessor is interrupted. This is a maskable interrupt and can be disabled. The 8085 has a nonmaskable and three additional vectored interrupt signals as well.

The 8085 interrupt process can be described in terms of eight steps:

Step 1: The interrupt process should be enabled by writing the instruction EI in the main program.

Instruction EI (Enable Interrupt)

- This is a 1-byte instruction.
- The instruction sets the Interrupt Enable flip-flop and enables the interrupt process.
- System reset or an interrupt disables the interrupt process.

Instruction DI (Disable Interrupt)

- This is a 1-byte instruction.
- The instruction resets the Interrupt Enable flip-flop and disables the interrupt process.
- It should be included in a program segment where an interrupt from an outside source cannot be tolerated.

Step 2: When the microprocessor is executing a program, it checks the INTR line during the execution of each instruction.

Step 3: If the line INTR is high and the interrupt is enabled, the microprocessor completes the current instruction, disables the Interrupt Enable flip-flop and sends a signal INTA – Interrupt Acknowledge (active low). The processor cannot accept any interrupt request until the interrupt flip-flop is enabled again.

Step 4: The signal INTA is used to insert a restart (RST) instruction (or a Call instruction) through external hardware. The RST instruction is a 1-byte call instruction that transfers the program control to a specific memory location.

- Step 5:** When the microprocessor receives an RST instruction, it saves the memory address of the next instruction on the stack. The program is transferred to the CALL location.
- Step 6:** Assuming that the task to be performed is written as a subroutine at the specified location, the processor performs the task. The subroutine is known as service routine.
- Step 7:** The service routine should include the instruction EI to enable the interrupt again.
- Step 8:** At the end of the subroutine, the RET instruction retrieves the memory address where the program was interrupted and continues the execution.

12.11 RST (Restart) Instructions

These are 1-byte Call instructions that transfer the program execution to a specific location. The RST instructions are executed in a similar way to that of Call instructions as listed in Table 12.1.

TABLE 12.1**Restart Instructions**

Mnemonics	Binary Code								Hex Code	Call Locations
	D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀		
RST 0	1	1	0	0	0	1	1	1	C7	0000
RST 1	1	1	0	0	1	1	1	1	CF	0008
RST 2	1	1	0	1	0	1	1	1	D7	0010
RST 3	1	1	0	1	1	1	1	1	DF	0018
RST 4	1	1	1	0	0	1	1	1	E7	0020
RST 5	1	1	1	0	1	1	1	1	EF	0028
RST 6	1	1	1	1	0	1	1	1	F7	0030
RST 7	1	1	1	1	1	1	1	1	FF	0038

In Figure 12.1, the instruction RST 5 is built using resistors and tri-state buffer.

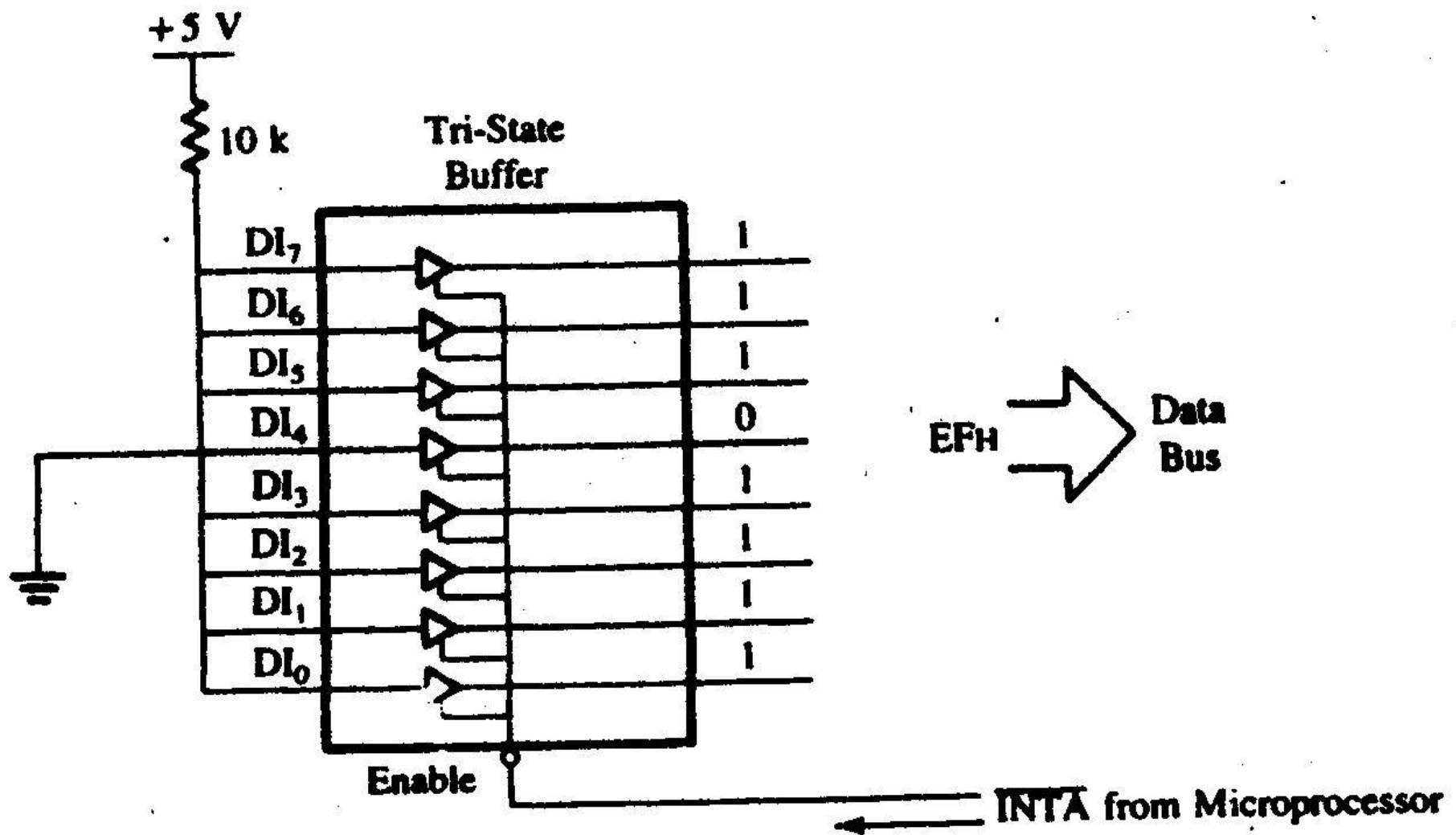


FIGURE 12.1
A Circuit to Implement the Instruction RST 5

12.1.2 Illustration: An Implementation of the 8085 Interrupt

PROBLEM SATEMENT

1. Write a main program to count continuously in binary with a one second delay between each count.
2. Write service routine at XX70H to flash FFH five times when the program is interrupted, with some appropriate delay between each count.

MAIN PROGRAM

LXI SP, XX99H
EI

NXTCNT: MVI A, 00H
OUT PORT1

MVI C, 01H
CALL DELAY

INR A
JMP NXTCNT

Service Routine

SERV: PUSH B
 PUSH PSW

 MVI B, 0AH

FLASH: MVI A, 00H
 OUT PORT1

 MVI C,01H
 CALL DELAY

 CMA

 DCR B
 JNZ FLASH

 POP PSW
 POP B
 EI
 RET

DESCRIPTION OF THE INTERRUPT PROCESS

1. The program will count continuously from 00H to FFH with a one second delay between each count.
2. To interrupt the processor, push the switch, the INTR line goes high.
3. Assuming the switch is pushed when the processor is executing the instruction OUT.
 - a.
 - b.
 - c.
 - d.
 - e. The program is transferred to memory location 0028H. The locations 0028-29-2AH should have the following Jump instructions to transfer the program to the service routine

JMP XX70H

(However, you do not have access to write at 0028H in the monitor program.
See the next section)

4. The programs jumps to the service routine at XX70H.
- 5.
- 6.
- 7

TESTING INTERRUPT ON A SINGLE-BOARD COMPUTER SYSTEM

In single-board microcomputers, some restart locations are usually reserved for users, and the system designer provides a Jump instruction at a restart location to jump somewhere in R/W memory. For example, in Intel's SDK-85 system, R/W memory begins at page 20H, and you may find the following instruction in the monitor program at memory location 0028H:

0028 JMP 20C2H

If instruction RST 5 is inserted as shown in Figure 12.3 , it transfers the program to location 0028H, and the monitor transfers the program from 0028H to location 20C2H. To implement the interrupt shown in Figure 12.3, you need to store the Jump instruction as shown below:

20C2 C3 JMP SERV
20C3 70
20C4 20

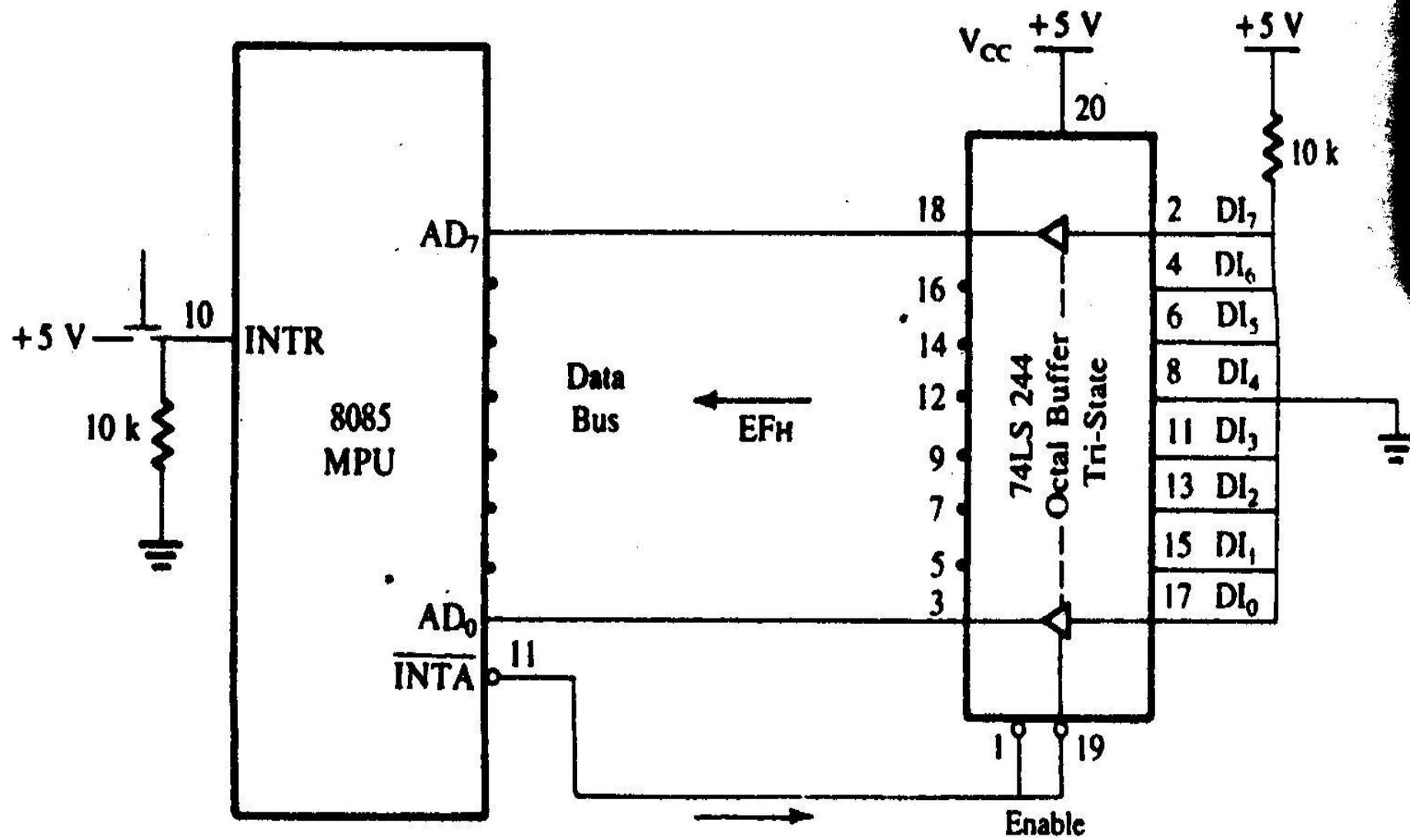


FIGURE 12.3
Schematic to Implement the 8085 Interrupt

ISSUES IN IMPLETING INTERRUPTS

In the above illustration, some questions remain unanswered:

1. Is there a minimum pulse width required for the INTR signal?

The microprocessor checks INTR, one clock period before the last T-states of an instruction cycle. In the 8085, the Call instructions require 18 T-states; therefore, the INTR pulse should be high at least for 17.5 T-states. In a system with 3 MHz clock frequency (such as the SDK-85 system), the input pulse to INTR should be at least 5.8 μ s long.

2. How long can the INTR pulse stay high?

The INTR pulse can remain high until the interrupt flip-flop is set by the EI instruction in the service routine. If it remains high after the execution of the EI instruction, the processor will be interrupted again, as if it were a new interrupt. In Figure 12.3, the manual push button will keep the INTR high for more than 20 ms; however, the service routine has a delay of 1 second, and the EI instruction is executed at the end of the service routine.

3. Can the microprocessor be interrupted again before the completion of the first interrupt service routine?

The answer to this question is determined by the programmer. After the first interrupt, the interrupt process is automatically disabled. In the Illustrative program in section 12.12, the service routine enables the interrupt at the end of the service routine; in this case, the microprocessor cannot be interrupted before the completion of this routine. If the instruction EI is written at the beginning of the routine, the microprocessor can be interrupted again during the service routine.

12.2 8085 VECTORED INTERRUPTS

The 8085 has five interrupt inputs (Figure 12.5). One is called INTR, three are called RST 5.5, RST 6.5, and RST 7.5 respectively, and the fifth is called TRAP, a nonmaskable interrupt. These last four (RSTS and TRAP) are automatically vectored (transferred) to a specific locations on memory page 00H without any external hardware. They do not require the INTA signal or an input port; the necessary hardware is already implemented inside the 8085. These interrupts and their call locations are as follows:

<u>Interrupts</u>	<u>Call Locations</u>
1. TRAP	0024H
2. RST 7.5	003CH
3. RST 6.5	0034H
4. RST 5.5	002CH

The TRAP has highest priority, followed by RST 7.5, 6.5, 5.5, and INTR, in that order; however, the TRAP has lower priority than the Hold signal used for DMA.

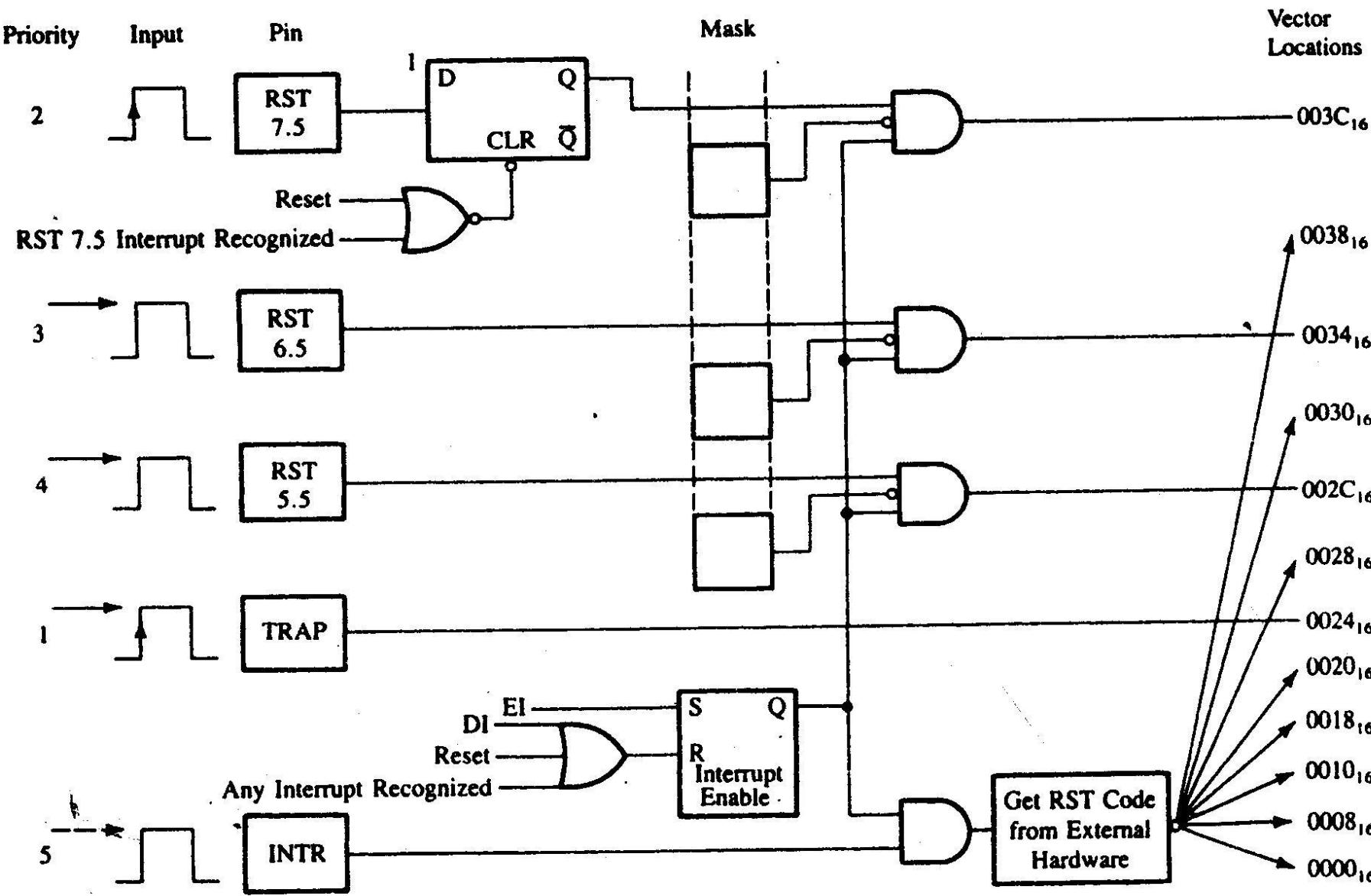


FIGURE 12.5

The 8085 Interrupts and Vector Locations

SOURCE: Intel Corporation, *MCS'80/85 Student Study Guide* (Santa Clara, Calif.: Author, 1979).

Property of
Seminar Library

12.2.1 TRAP

TRAP, a nonmaskable interrupt known as NMI, is analogous to smoke detector. It has the highest priority among the interrupt signals, it need not to be enabled, and cannot be disabled. It is level-and edge-sensitive, meaning that the input should go high to be acknowledged. It cannot be acknowledged again until it makes transition from high to low to high.

Figure 12.5 shows that when this interrupt is triggered, the program control is transferred to location 0024H without any external hardware or the interrupt enable instruction EI. TRAP is generally used for such critical events as power failure and emergency shut-off.

12.2.2 RST 7.5, 6.5, and 5.5

These maskable interrupt are enabled under program control with two instructions: EI (Enable Interrupt), and SIM (Set Interrupt Mask) described below:

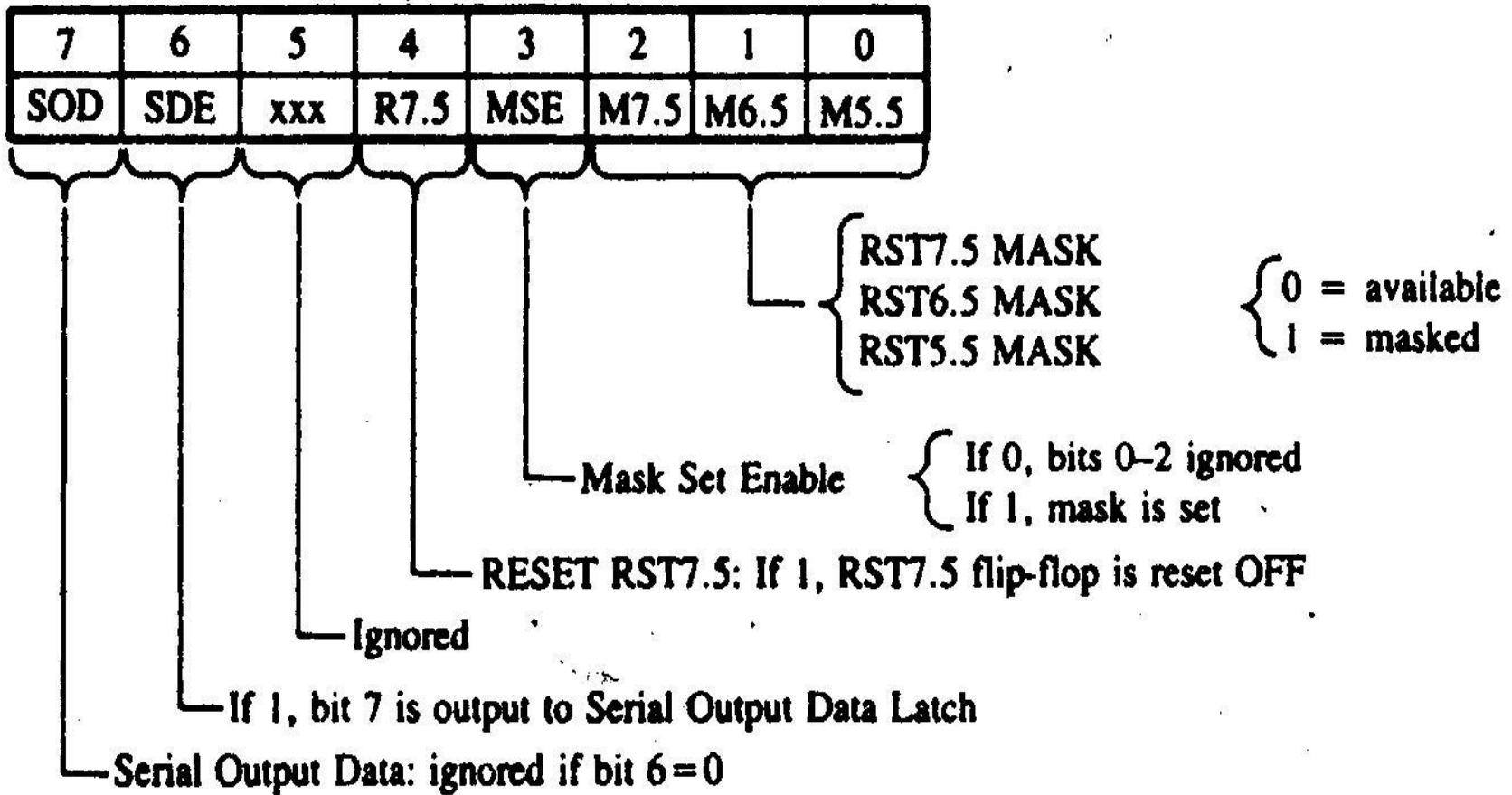


FIGURE 12.6

Interpretation of the Accumulator Bit Pattern for the SIM Instruction

SOURCE: Intel Corporation, *Assembly Language Programming Manual* (Santa Clara, Calif.: Author, 1979), pp. 3-59.

Instruction SIM:

Set Interrupt Mask. This is a 1-byte instruction and can be used for three different functions (Figure 12.6).

- One function is to set for RST 7.5, 6.5, and 5.5 interrupts. This instruction reads the content of the accumulator and enables or disables the interrupts according to the contents of accumulator. Bit D₃ is a control bit and should = 1 for bits D₀, D₁, and D₂ to be effective. Logic 0 on D₀, D₁, and D₂ will enable the corresponding interrupts, and logic 1 will disable the interrupts.
- The second function is to reset RST 7.5 flip-flop (Figure 12.6). Bit D₄ is additional control for RST 7.5. If D₄=1, RST 7.5 is reset. This is used to override (or ignore) RST 7.5 without servicing it.
- The third is to implement serial I/O (Bits D₇ and D₆) and do not affect the interrupts.

The entire interrupt process (except TRAP) is disabled by resetting the Interrupt Enable flip-flop (Figure 12.5). The flip-flop can be reset in one of the three ways:

- Instruction DI
- System Reset
- Recognition of an Interrupt Request

Example: 12.1: Enable all the interrupts in an 8085 system.

Instruction: EI ; Enable interrupts
MVI A, 08H ; To enable RST 7.5, 6.5, and 5.5
SIM ; Enable RST 7.5, 6.5, and 5.5

Example: 12.2: Reset the 7.5 interrupt from Example 12.1.

Instruction: MVI A, 18H ; Set D₄ = 1
SIM ; Reset 7.5 interrupt flip-flop

PENDING INTERRUPTS

When one interrupt request is being served, other interrupt request may occur and remain pending. The 8085 has an additional instruction call RIM (Read Interrupt Mask) to sense these pending interrupts (Figure 12.7).

Instruction RIM:

Read Interrupt Mask. This is a 1-byte instruction that can be used for the following functions:

The RIM instruction loads the accumulator with the following information:

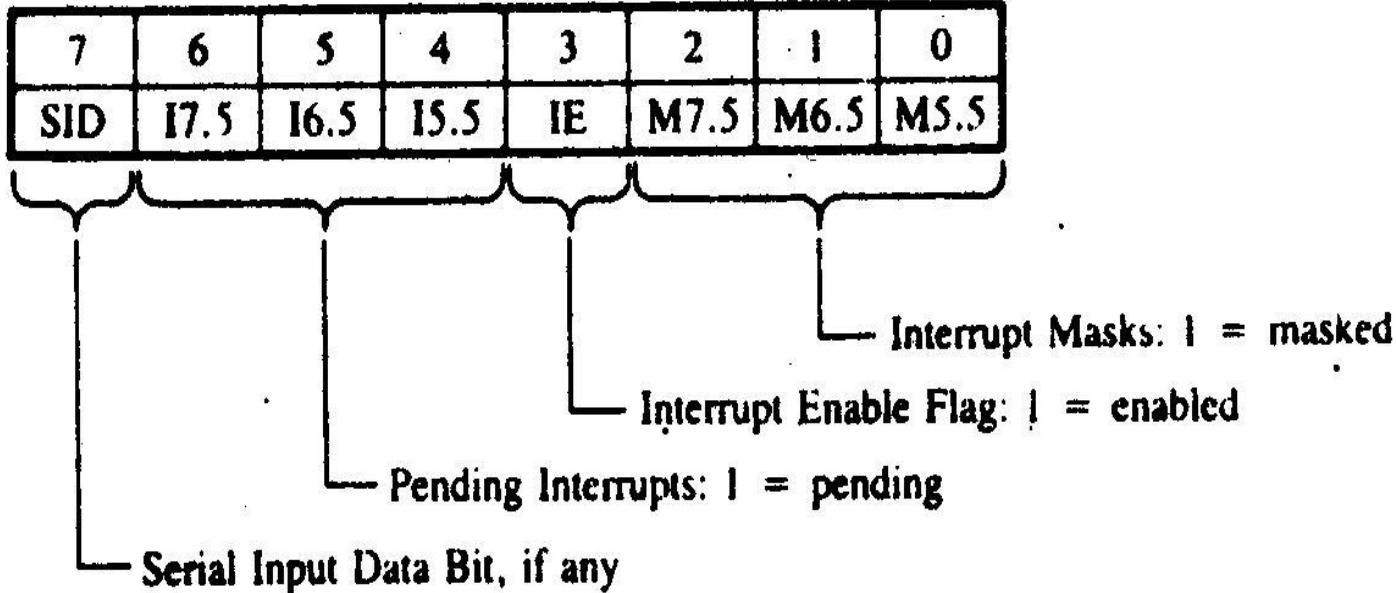


FIGURE 12.7

Interpretation of the Accumulator Bit Pattern for the RIM Instruction

SOURCE: Intel Corporation. *Assembly Language Programming Manual* (Santa Clara, Calif.: Author, 1979), pp. 3-49.

- ❑ To read interrupt masks, This instruction loads the accumulator with 8 bits indicating the current status of the interrupt masks (Figure 12.7).
- ❑ To identify pending interrupts. Bits D₄, D₅, and D₆ (Figure 12.7) identify the pending interrupts.
- ❑ To receive serial data bit D₇ is used (Figure 12.7).

Example: 12.3: Assuming the microprocessor is completing an RST 7.5 interrupt request, check to see if RST 6.5 is pending. If it is pending, enable RST 6.5 without affecting any other interrupts; otherwise return to the main program.

Instruction:

RIM	
MOV B, A	
ANI 20H	; Check whether RST 6.5 is pending
JNZ NEXT	
EI	
RET	
MOV A,B	
ANI 0DH	; Enable RST 6.5 by sending D ₁ =0
ORI 08H	; Enable SIM by setting D ₃ =1
SIM	
JMP SERV	; Jump to service routine for RST 6.5

12.2.3 Illustration: Interrupt-Driven Clock

PROBLEM STATEMENT

Design a 1-minute timer using a 60 H_z power line as an interrupting source. The output ports should display minutes and seconds in BCD. At the end of the minute, the output ports should continue displaying one minute and zero seconds.

HARDWARE DESCRIPTION

This 1-minute timer is designed with a 60 H_z AC line. The circuit (Figure 12.8) uses a step-down transformer; the 74121 monostable multivibrator, and interrupt pin RST 6.5. After the interrupt the program control is transferred to memory location 0034H in the monitor program.

The AC line with 60 H_z frequency has a period of 16.6 ms; that means it can provide a pulse every sixtieth of a second with 8.3 ms pulse width, which is too long for the interrupt. The interrupt flip-flop is enabled again before 6 μs in the service routine, therefore the pulse should be turned off before the EI instruction in service routine is executed.

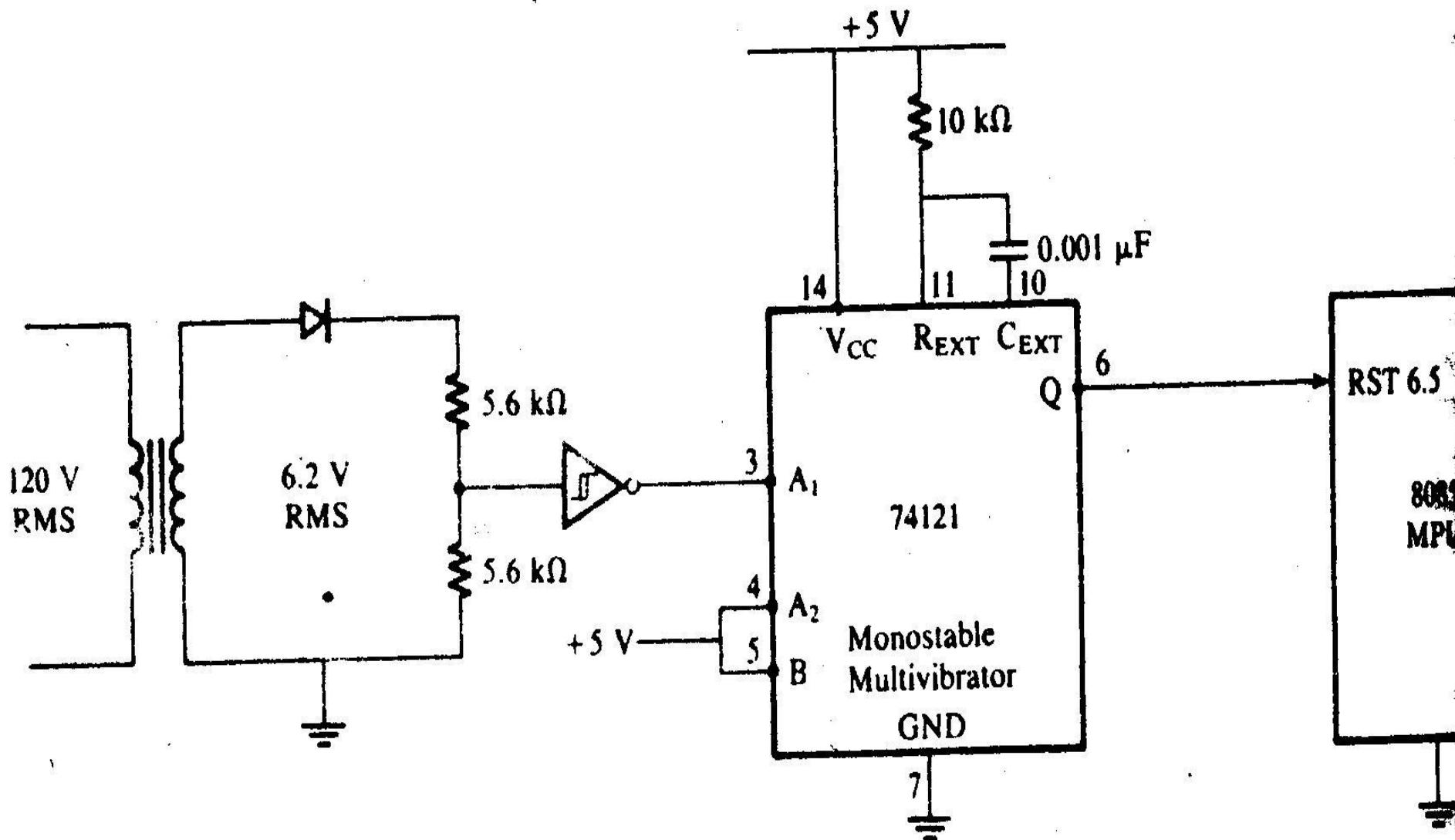


FIGURE 12.8

Schematic of Interrupt-Driven Timer Clock

Monitor Program

0034 JMP RWM

Main Program

 LXI SP, STACK
 RIM
 ORI 08H
 SIM
DISPLAY: LXI B,0000H ; Set up B for minutes and C for seconds

 MVI D, 3CH ; Set up D to count 60_{10} interrupts
 EI

DISPLAY: MOV A, B
 OUT PORT1
 MOV A, C
 OUT PORT2
 JMP DISPLAY

RWM: JMP TIMER ; This is RST 6.5 vector location 0034H

Interrupt Service Routine

; Section I

TIMER: DCR D ;One interrupt occurred

 EI

 RNZ

; Has 1 second elapsed? If not, return

;Section II

 DI

 MVI D,3CH

;1 second is complete; load D to count again

 MOV A,C

 ADD 01H

 DAA

; Decimal-adjust “Second”

 MOV C,A

;Save “BCD” seconds

 CPI 60H

 EI

 RNZ

;Section III

 DI

 MVI C,00H

;60 seconds complete; clear “Seconds” register

 INR B

; Increment “Minutes” register

 RET

Chapter-13

Interfacing Data Converters

13.1 DIGITAL-TO-ANALOG (D/A) CONVERTERS

Digital-to-Analog converters can be classified in three categories:

- current output: provides current as output signal.
- voltage output: internally converts current signal into voltage signal.
- multiplying type: output current or voltage is product of input current or voltage.

Voltage output is slower than current output because of the delay in converting the current signal into voltage signal. However, in many applications, it is necessary to convert current into voltage by using an external operational amplifier.

13.1.1 Basic Concepts

Figure 13.1(a) shows a block diagram of a 3-bit D/A converter. If the input ranges from 0 to 1 V, it can be divided into eight equal parts ($1/8$ V); each successive is $1/8$ V higher than the previous combination, as shown in Figure 13.1(b).

The following points can be summarized from the graph:

1. If a converter has n input lines, it can have 2^n input combinations.
2. If the full-scale analog voltage is 1 V, the smallest unit or the LSB (001_2) is equivalent to $1/2^n$ of 1 V. This is defined as resolution.
3. The MSB represents half of the full-scale value. In this example, the MSB (100_2) = $1/2$ V.
4. For the maximum input signal (111_2), the output signal is equal to the value of the full scale input signal minus the value of the 1 LSB input signal. In this example, the maximum input signal (111_2) represents $7/8$ V.

Example 13.1: Calculate the values of the LSB, MSB, and full-scale output for an 8-bit DAC for the 0 to 10 V range.

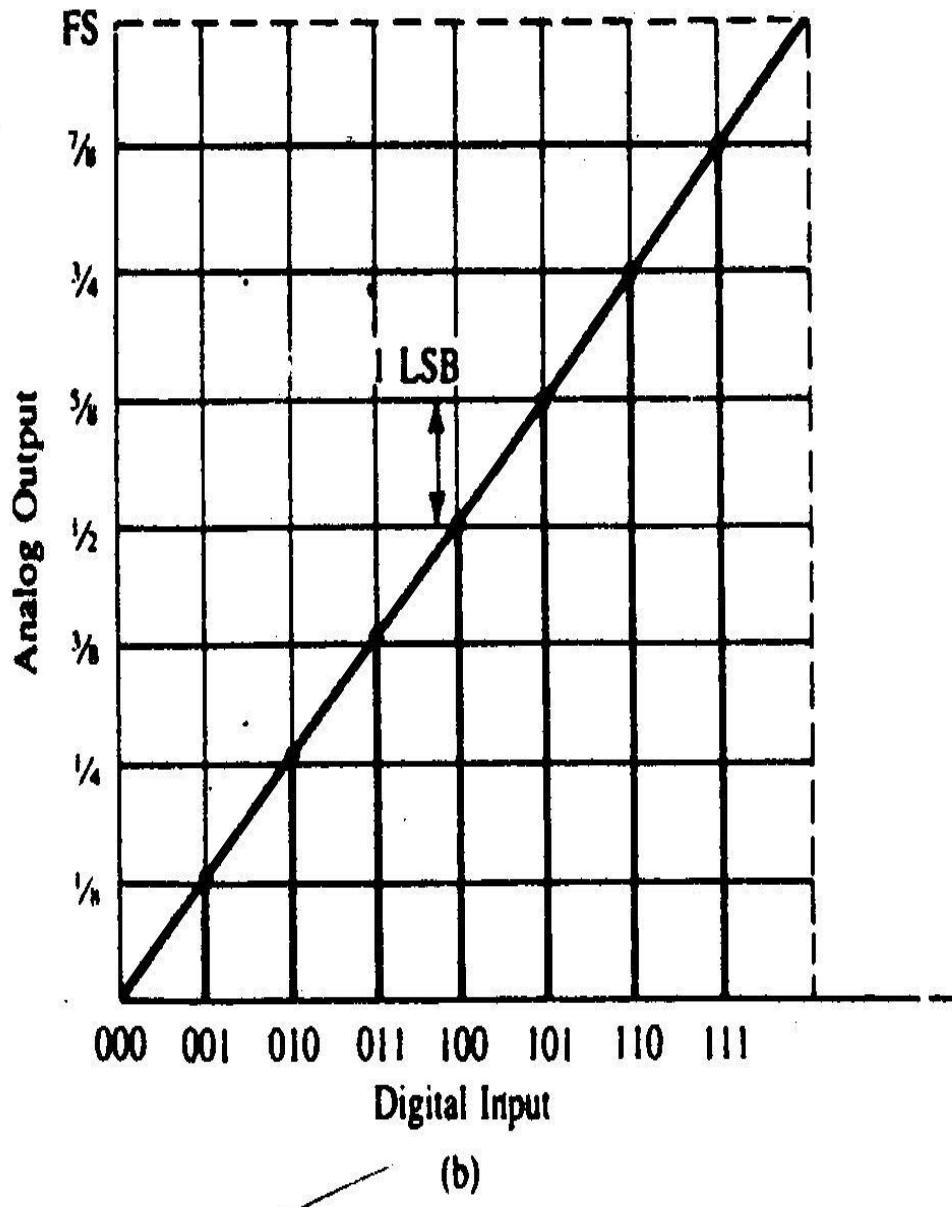
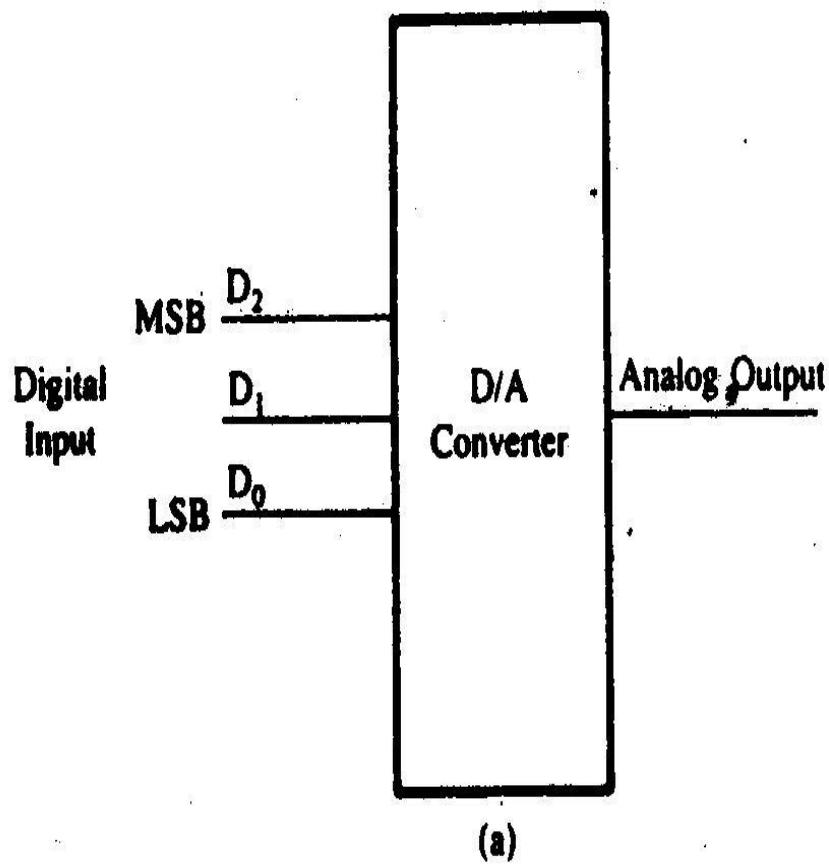


FIGURE 13.1

A 3-Bit D/A Converter: Block Diagram (a) and Digital Input vs. Analog Output (b)

Solution:

$$1. \text{ LSB} = 1/2^8 = 1/256$$

$$\text{For } 10 \text{ V, LSB} = 10 \text{ V}/256 = 39 \text{ mV}$$

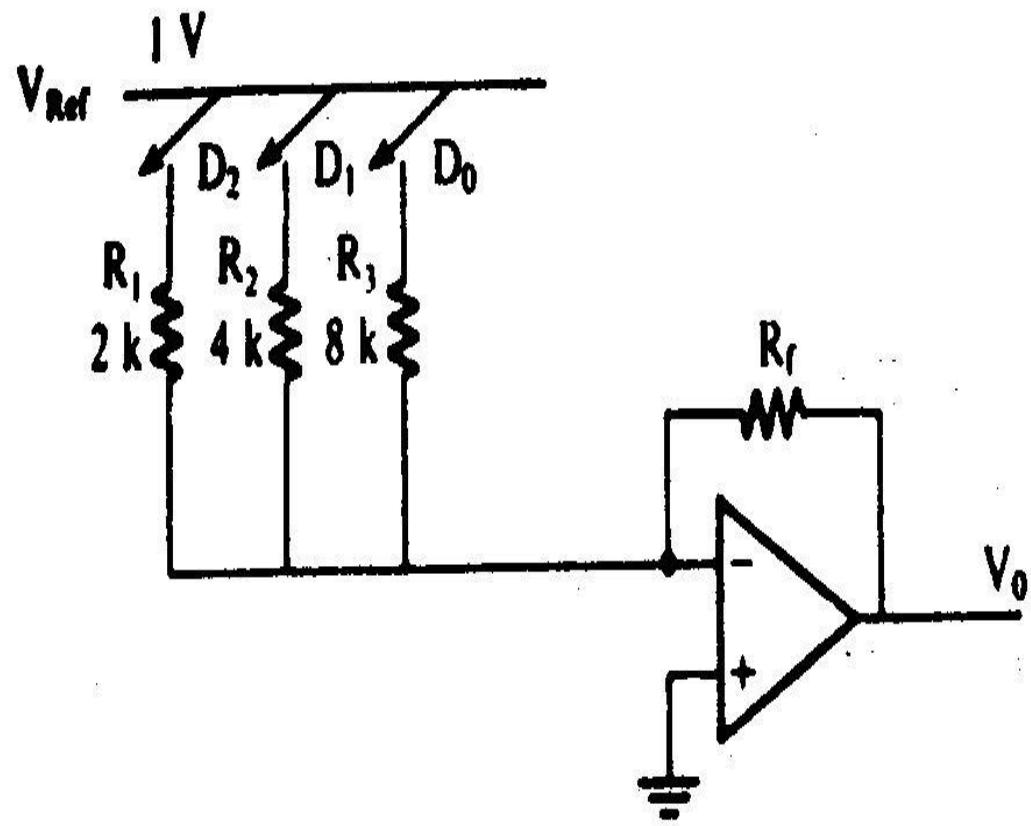
$$2. \text{ MSB} = 1/2 \text{ full scale} = 5\text{V}$$

$$\begin{aligned}3. \text{ Full-Scale Output} &= (\text{Full-Scale Value} - 1 \text{ LSB}) \\&= 10\text{V} - 0.039 \text{ V} \\&= 9.961 \text{ V}\end{aligned}$$

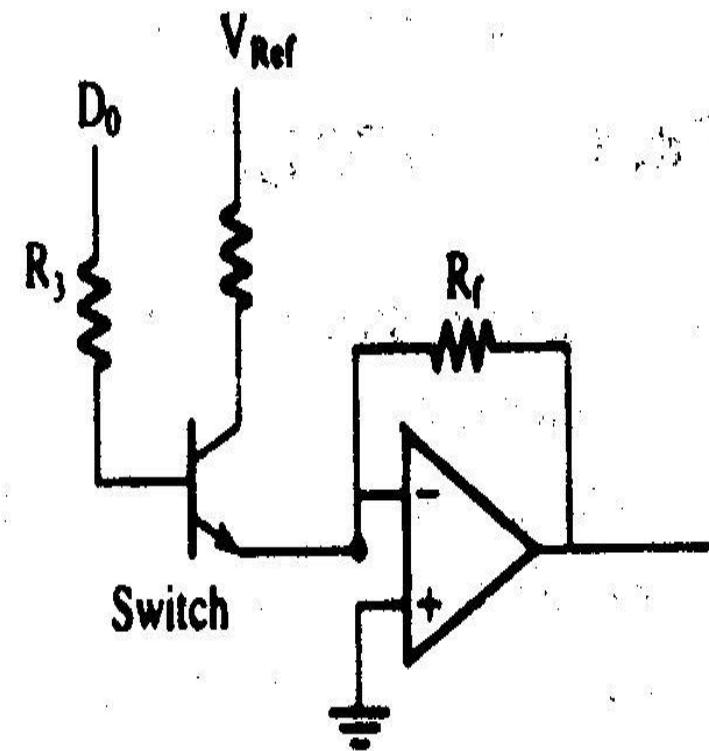
13.1.2 D/A Converter Circuits

The input resistors R_1 , R_2 , and R_3 are selected in binary weighted proportion; each has double the value of the previous resistor. If all three inputs are 1 V, the total output current is

$$\begin{aligned}I_o &= I_T = I_1 + I_2 + I_3 \\&= V_{in}/R_1 + V_{in}/R_2 + V_{in}/R_3 \\&= V_{in}/1\text{k}(1/2 + 1/4 + 1/8) \\&= 0.875 \text{ mA}\end{aligned}$$



(a)



(b)

FIGURE 13.3

Simulated D/A Converter (a) and Transistor Switch to Turn On/Off Bit D_0 (b)

The voltage output is

$$\begin{aligned}V_o &= - R_f I_T \\&= - (1 \text{ k}) (0.875 \text{ mA}) \\&= |7/8|\end{aligned}$$

This example shows that for the input = 111_2 , the output is equal to either $7/8$ mA or $7/8$ V, representing the D/A conversion process.

Now the output current I_o can be generalized for any number of bits as

$$I_o = V_{\text{Ref}} / R (A_1/2 + A_2/4 + \dots + A_n/2^n)$$

The following points can be inferred from the above example:

1. A D/A converter circuit requires three elements: resistor network with appropriate weighting, switches, and a reference source.
2. The output can be a current signal or converted into a voltage signal using an operational amplifier.
3. The time required for conversion, called settling time, is dependent on the response of the switches and output amplifier (for a voltage output DAC).

13.1.3 Illustration: Interfacing an 8-Bit D/A Converter

PROBLEM STATEMENT

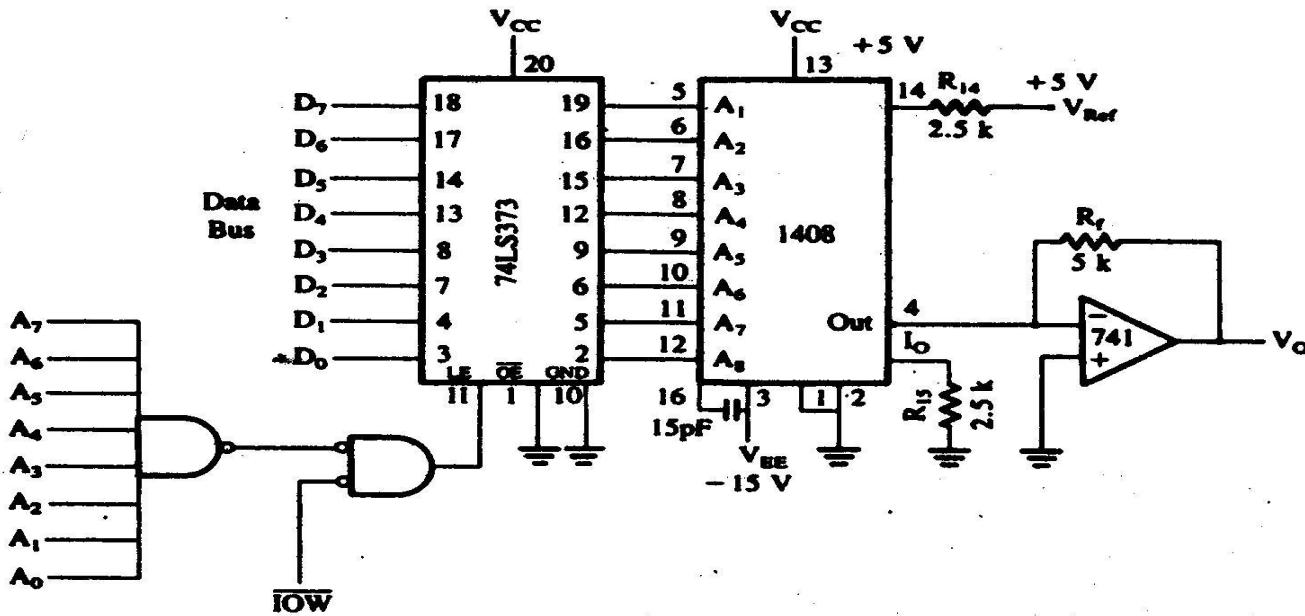
1. Design an output port with the address FFH to interface the 1408 D/A converter that is calibrated for a 0 to 10 V range; refer to Figure 13.5(a).
2. Write a program to generate a continuous ramp waveform.
3. Explain the operation of the 1408 in Figure 13.5(b) , which is calibrated for a bipolar range $\pm 5V$. Calculate the output V_O if the input is 10000000_2 .

HARDWARE DESCRIPTION

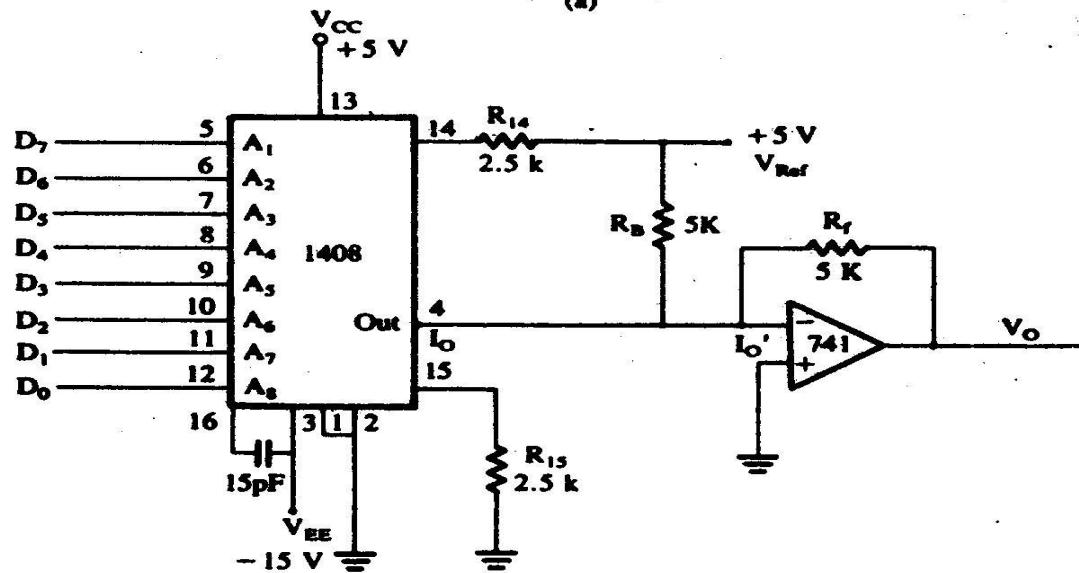
The total reference current source is determined by the resistor R_{14} and the voltage V_{Ref} . The resistor R_{15} is generally equal to R_{14} to match the input impedance of the reference source. The output I_O is calculated as follows:

$$I_O = V_{Ref} / R_{14} (A_1/2 + A_2/4 + A_3/8 + A_4/16 + A_5/32 + A_6/64 + A_7/128 + A_8/256)$$

Where inputs A_1 through $A_8 = 0$ or 1 .



(a)



(b)

FIGURE 13.5
Interfacing the 1408 D/A Converter: Voltage Output in Unipolar Range (a) and in Bipolar Range (b)

This formula is an application of the generalized formula for the current I_o . For full scale input (D_7 through $D_0 = 1$).

$$\begin{aligned}I_o &= 5 \text{ V} / 2.5 \text{ k} (1/2 + 1/4 + 1/8 + 1/16 + 1/32 + 1/64 + 1/128 + 1/256) \\&= 2 \text{ mA} (255/256) \\&= 1.992 \text{ mA.}\end{aligned}$$

The output is 1 LSB less than full-scale reference source of 2 mA. The output voltage V_o for the full-scale input is

$$\begin{aligned}&= 2 \text{ mA} (255/256) \times 5 \text{ k} \\&= 9.961 \text{ V}\end{aligned}$$

PROGRAM

To generate a continuous waveform, the instructions are as follows:

	MVI A, 00H	; First input 00H for DAC
DTOA:	OUT FFH	; Output to DAC
	MVI B, COUNT	; COUNT for delay
DELAY:	DCR B	
	JNZ DELAY	
	INR A	; Next input for DAC
	JMP DTOA	

Program Description:

The program outputs 00H to FFH as the digital inputs for DAC. The analog output of the DAC starts at 0 V and increases up to 10 V (approximately) as a ramp. When the accumulator contents go to 0, the next cycle begins; thus the ramp signal is generated continuously and that can be displayed on oscilloscope.

The delay in the program is necessary for two reasons:

1. The time needed for a microprocessor to execute an output loop is likely to be less than the settling time of the DAC.
2. The slope of the ramp can be varied by changing the delay.

OPERATING THE D/A CONVERTER IN A BIPOLEAR RANGE

The 1408 in Figure 13.5(b) is calibrated for the range from -5 V to +5 V by adding the resistor R_B (5.0 k) between the reference voltage V_{Ref} and the output pin. The resistor R_B supplies 1 mA (V_{Ref} / R_B) current to the opposite direction of the current generated by the input signal. Therefore, the output current for the bipolar operation I_o is

$$\begin{aligned}I_o' &= I_o - V_{Ref} / R_B \\&= V_{Ref} / R_{14} (A_1/2 + A_2/4 + A_3/8 + A_4/16 + A_5/32 + A_6/64 + A_7/128 + A_8/256) - V_{Ref} / R_B\end{aligned}$$

When the input signal is equal to zero, the output V_o is

$$\begin{aligned}V_o &= I_o' R_f \\&= (I_o - V_{Ref} / R_B) R_f \\&= (0 - 5 \text{ V}/5\text{k}) (5\text{k}) \quad (I_o = 0 \text{ for input} = 0) \\&= -5\text{V}\end{aligned}$$

When the input = 1000 0000, the output V_o is

$$\begin{aligned}V_o &= (I_o - V_{\text{Ref}} / R_B) R_f \\&= (V_{\text{Ref}} / R_{14} \times A_1/2 - V_{\text{Ref}} / R_B) R_f \quad (A_2 - A_8 = 0) \\&= (5 \text{ V}/2.5 \text{ k} \times 1/2 - 5 \text{ V}/5 \text{ k})5\text{k} \\&= (1 \text{ mA} - 1 \text{ mA})5 \text{ k} \\&= 0\end{aligned}$$

13.2 ANALOG-TO-DIGITAL CONVERTERS

The A/D conversion is a quantization process whereby an analog signal is represented by equivalent binary states; this is opposite to the D/A conversion process. Analog-to-Digital converters can be classified in two general groups based on the conversion technique:

- Successive-approximation A/D converter: used in instrumentation.
- Integration-type A/D converter: used in digital meters, monitoring systems.

13.2.1 Basic Concepts

Figure 13.8(a) shows a block diagram of a 3-bit A/D converter. Figure 13.8(b) shows the graph of the analog input voltage (0 to 1 V) and the corresponding digital output signal. The resolution of the converter is $1/8$ V.

13.2.1 Successive-Approximation A/D Converter

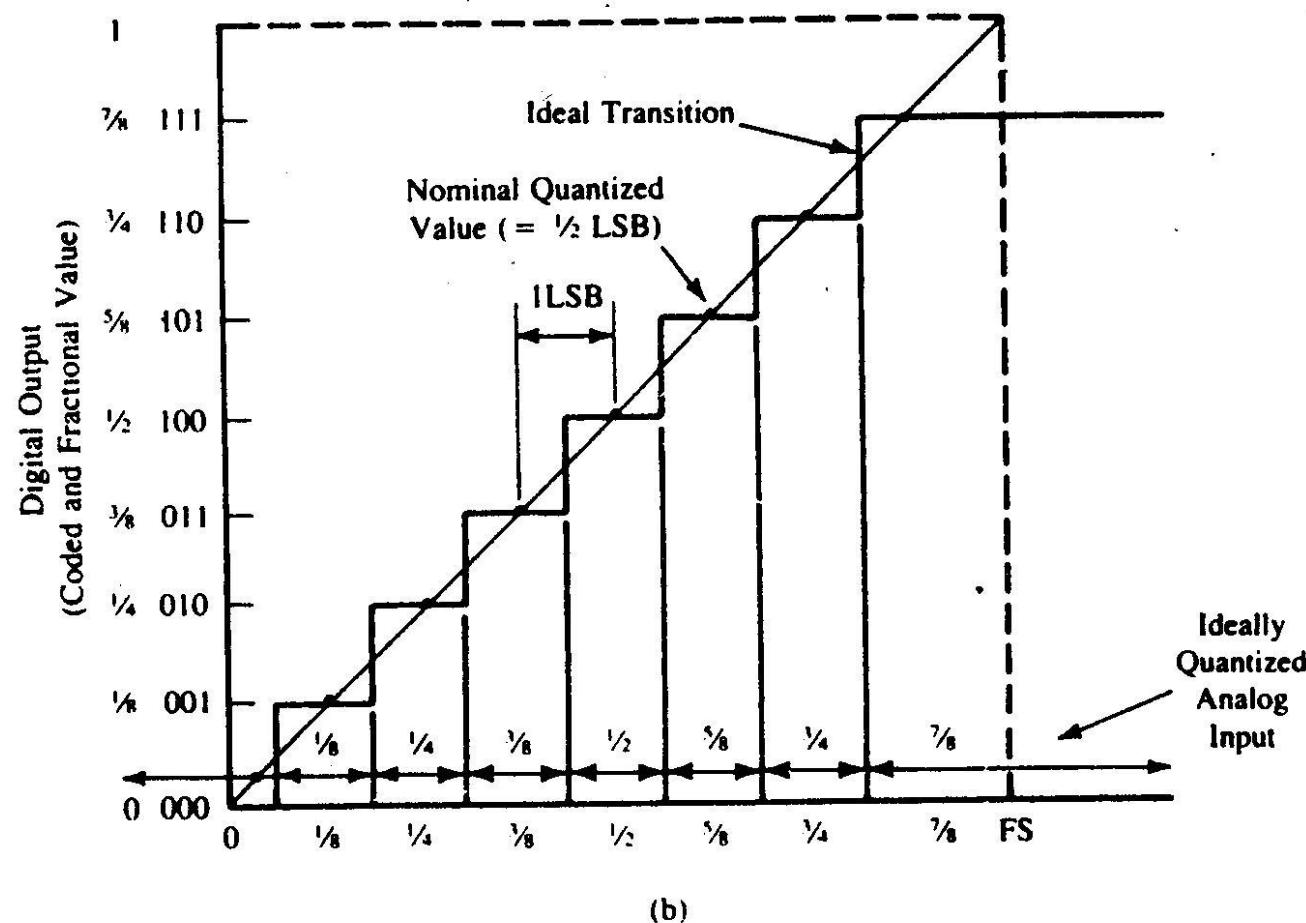
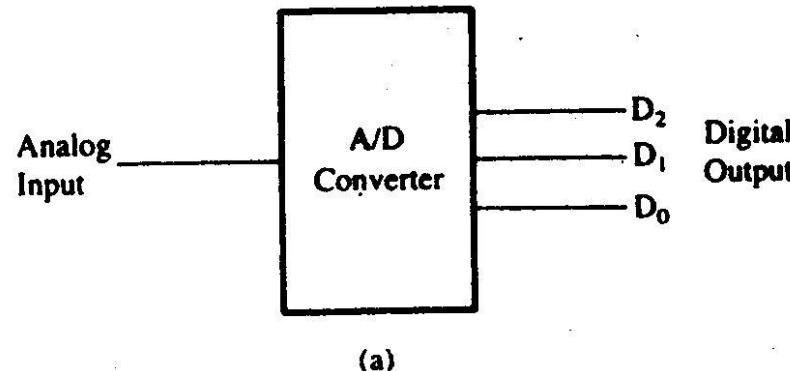
Figure 13.9(a) shows the block diagram of a successive approximation A/D converter includes three major elements: the D/A converter, the successive approximation register (SAR), and the comparator. The conversion technique involves the output of the D/A converter V_o with the analog input signal V_{in} . The digital input to the DAC is generated using the successive-approximation method (explain below). When the DAC output matches the analog signal, the input to the DAC is equivalent digital signal.

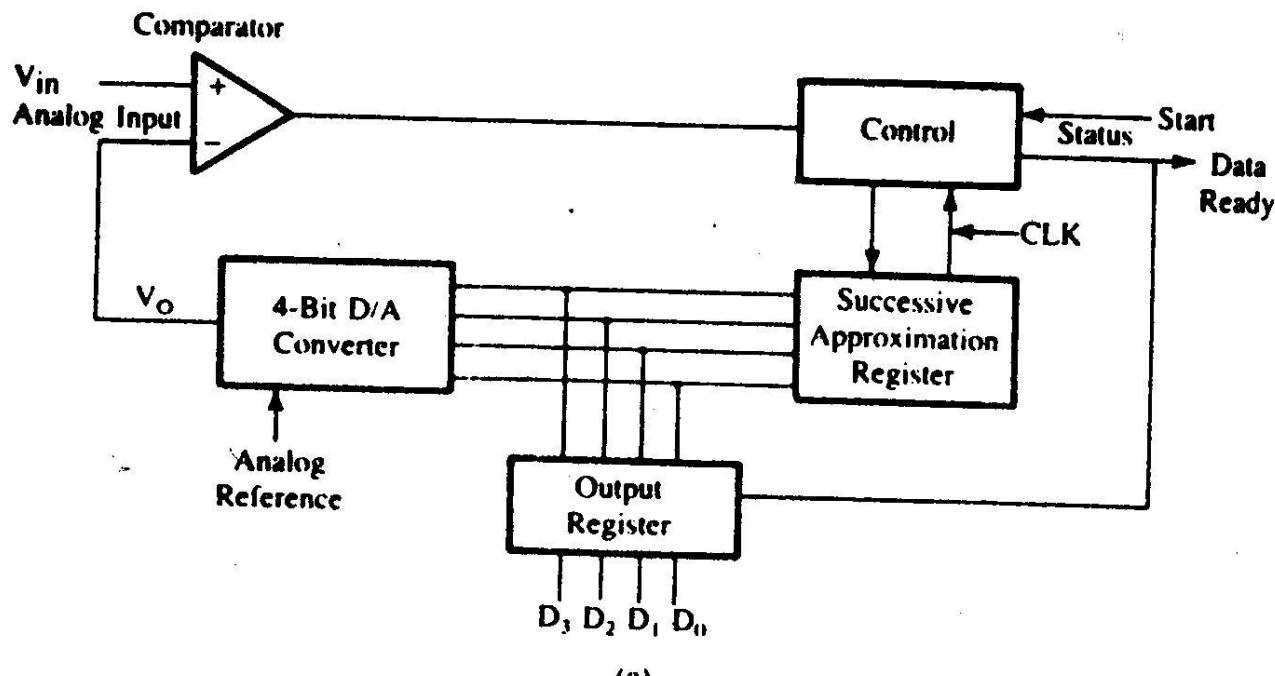
In the case of a 4-bit A/D converter, bit D_3 is turned on first and the output of the DAC is compared with an analog signal. If the comparator changes the state, indicating that the output generated by D_3 is larger than the analog signal, bit D_3 is turned off in the SAR and bit D_2 is turned on. The process continues until the input reaches bit D_0 .

FIGURE 13.8

A 3-Bit A/D Converter: Block
Diagram (a) and Analog Input vs.
Digital Output (b)

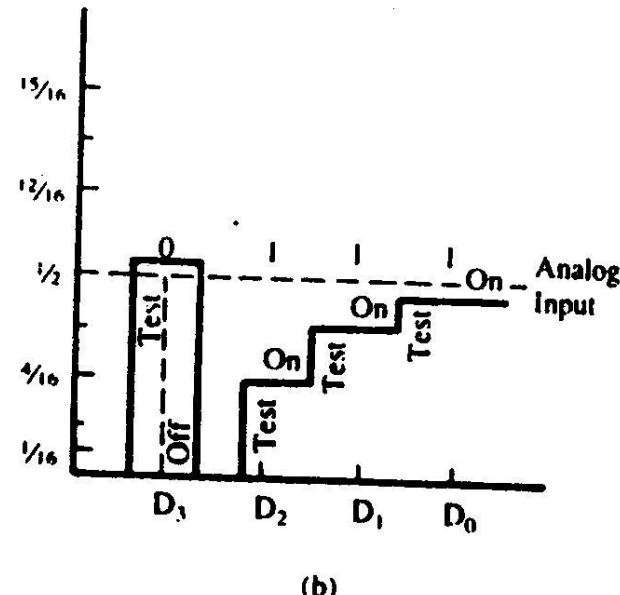
SOURCE: Analog Devices, Inc., *Integrated
Circuit Converters, Data Acquisition Systems,
and Analog Signal Conditioning Components*
(Norwood, Mass.: Author, 1979), pp. 1-18.





(a)

FIGURE 13.9
Successive-Approximation A/D
Converter: Block Diagram (a) and
Conversion Process for a 4-Bit
Converter (b)



(b)

Figure 13.9(b) illustrates a 4-bit conversion process. When bit D_3 is turned on, the output exceeds the analog signal and therefore, D_3 is turned off. When the next three successive bits are turned on, the output becomes approximately equal to the analog signal.

13.2.1 Interfacing 8-Bit A/D Converters

INTERFACING AN 8-Bit A/D CONVERTER USING STATUS CHECK

Figure 13.11 shows a schematic of interfacing a typical A/D converter using status check.

PROGRAM

```
OUT 82H  
TEST: IN 80H  
      RAR  
      JC TEST  
      IN 81H  
      RET
```

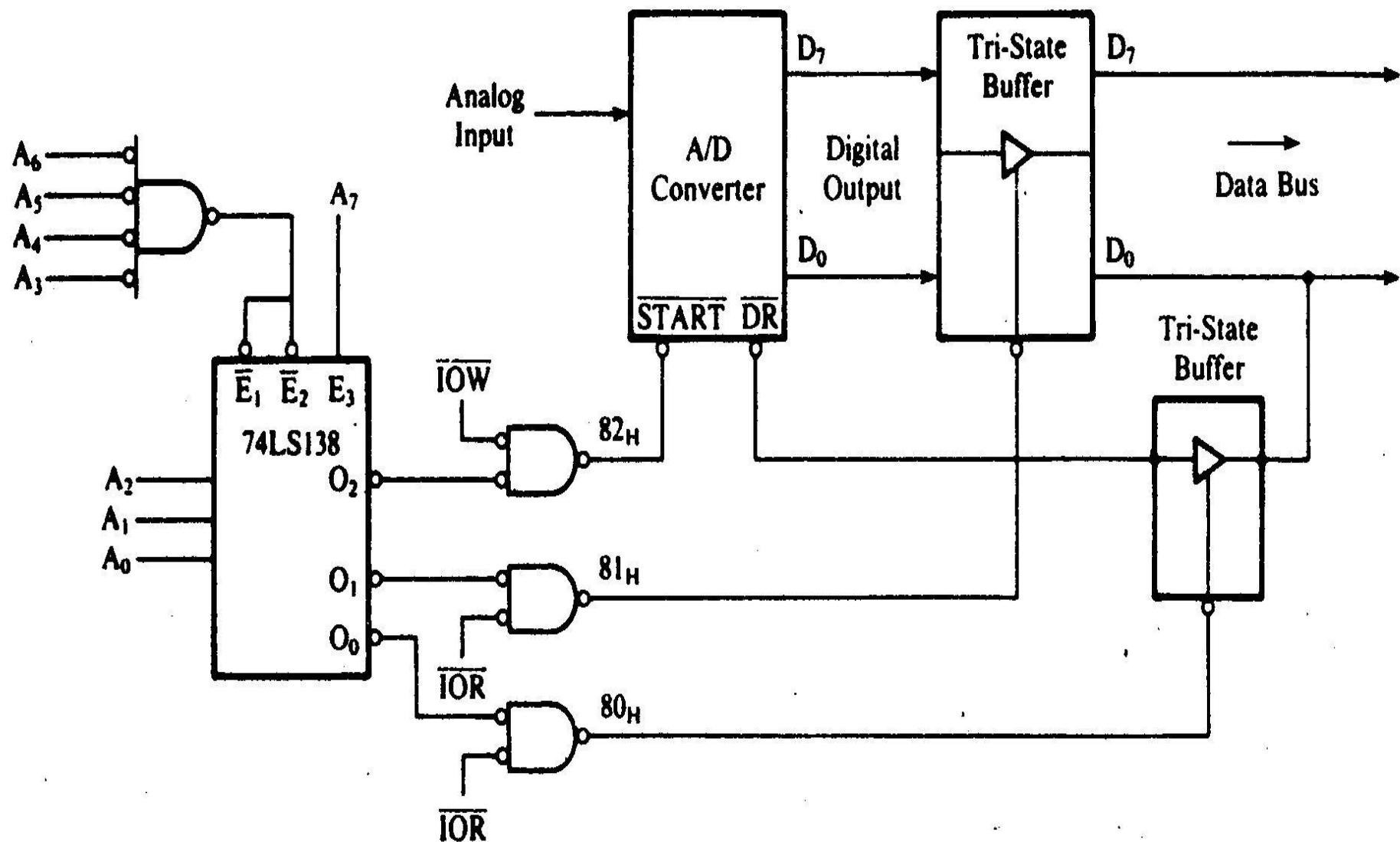


FIGURE 13.11
Interfacing an A/D Converter Using the Status Check

Chapter-14

Programmable Interface Devices:

- 8155 I/O and Timer**
 - 8279 Keyboard/Display Interface**
-

A programmable interface device is designed to perform various input/output functions. Such a device can be set up to perform specific functions by writing an instruction (or instructions) in its internal register, called **control register**.

14.1 BASIC CONCEPTS IN PROGRAMMABLE DEVICES

In Chapter 5 we assumed that the I/O devices were always ready for data transfer. But in the situation to interface a printer (a slower device) to a microprocessor (the fastest unit) there must be some signals exchanged between them prior to sending and receiving data. These signals are called handshake signals. Otherwise, processor will read the same data again and again for a slow responding input device and overwrite the previous data by the next data for a slow responding output device.

14.1.1 Making the 74LS245 Transceiver Programmable

The 74LS245 is a bidirectional tri-state octal buffer, and the direction of data flow is determined by the signal of DIR. When DIR is high, data flow from S to B, and when it is low, data flow from B to A.

In fact, this is a hard-wired programmable device; the direction of the data flow is programmed through DIR. However, we are interested in a device that can be programmed by writing an instruction through the MPU. This can be accomplished by adding a register called the control register, as shown in Figure 14.2, and by connecting the DIR signal to bit D_o of the control register. When $D_o = 1$, data flow from A to B as output, and when $D_o = 0$, data flow in the opposite direction as input.

Example 14.1: Write instructions to initialize the hypothetical chip (Figure 14.2) as an output buffer.

Solution: **Instructions**

MVI A,01H
OUT FFH
MVI A, BYTE1
OUT FEH

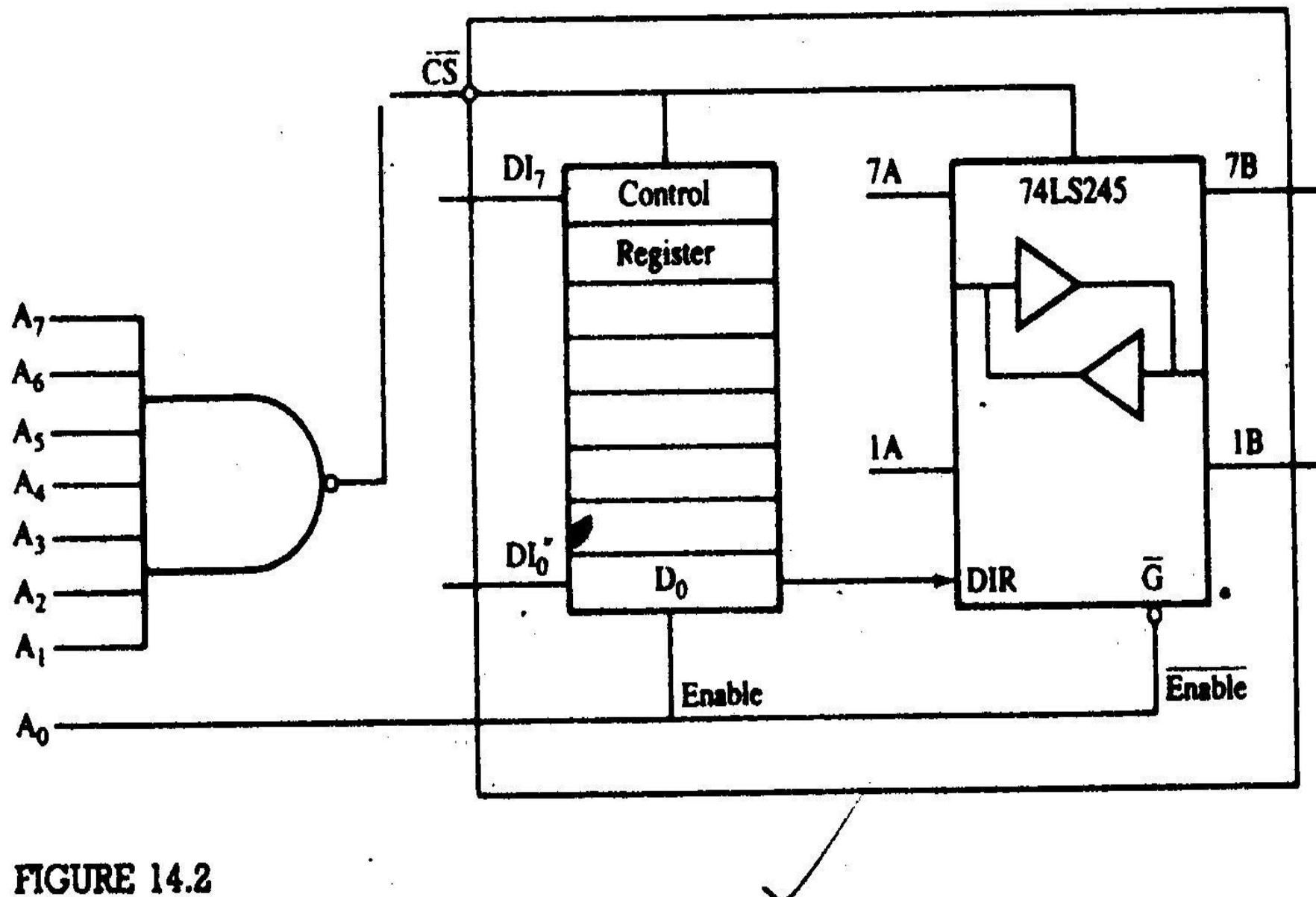


FIGURE 14.2
Making 74LS245 Programmable

14.1.2 Programmable Device with a Status Register

Figure 14.3 shows a hypothetical programmable device with a status register. There are three registers and can be accessed as ports. The port addresses and the functions of these registers can be summarized as follows:

Control register (output only) = FFH (A_1 and $A_0 = 1$)

Status register (input only) = FFH (A_1 and $A_0 = 1$)

Data register (input or output) = FFH ($A_1=1$, $A_0 = 0$)

14.1.3 Programmable Device with a Handshake Signals

The MPU and peripherals operate at different speeds; therefore, signals are exchanged prior to data transfer between fast-responding MPU and slow-responding peripherals such as printers and data converters. These signals are called **handshake signals**. These signals are generally provided by programmable devices.

Figure 14.4(a) shows a programmable device in the input mode, with two handshake signals: STB (Strobe) and IBF (Input Buffer Full) and one interrupt signal (INTR). Figure 14.4(b) shows the programmable device in the output mode using the same handshake signals, except that they are labeled differently: OBF (Output Buffer Full) and ACK (Acknowledgement).

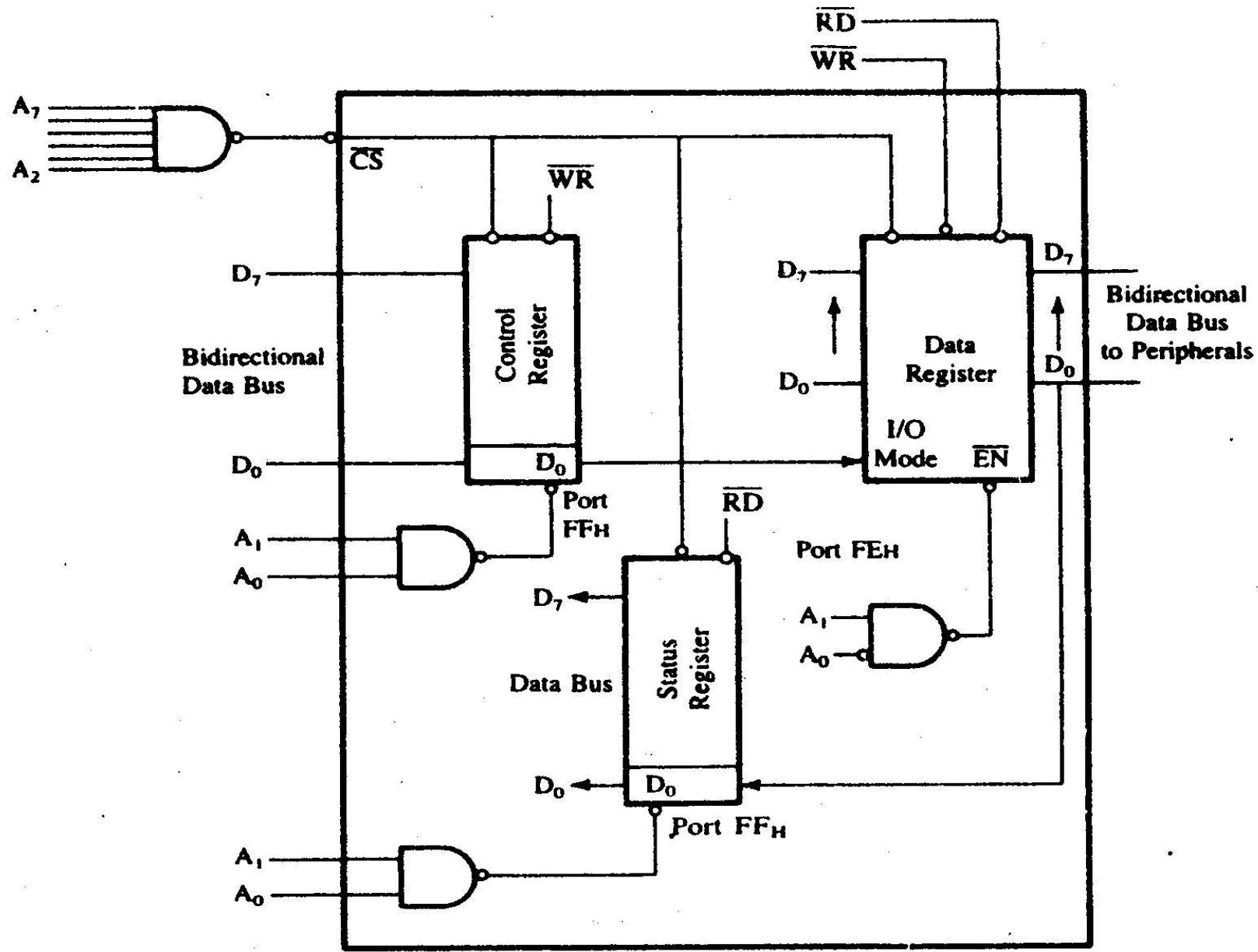


FIGURE 14.3
A Hypothetical Programmable Device with a Status Register

•• ANU AVA O TCUUS WIE BYW BY SCHUNING COMUOI SIGNAL RD.

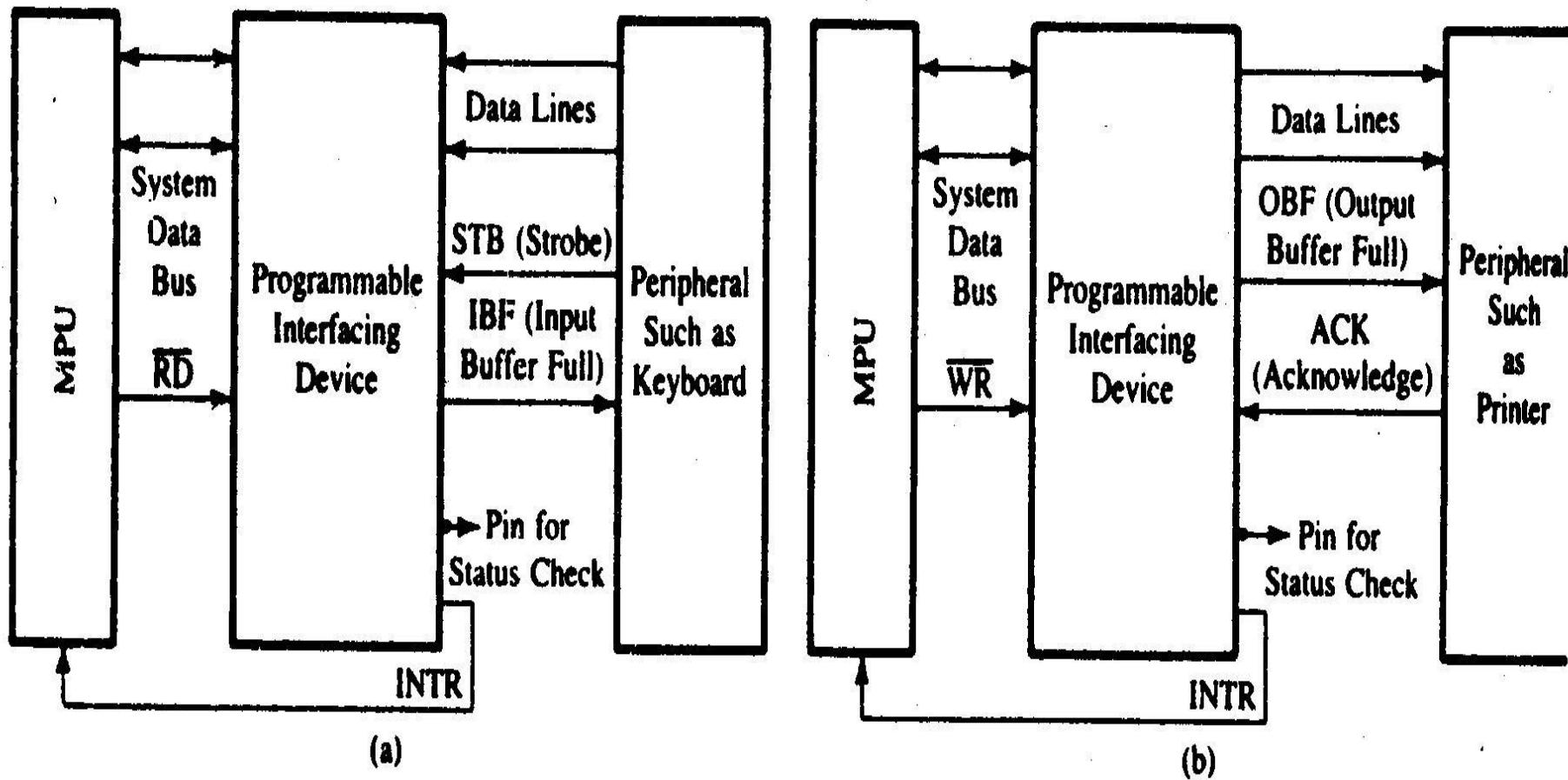


FIGURE 14.4
Interfacing Device with Handshake Signals for Data Input (a) and Data Output (b)

Examination of Figure 14.4 shows the following similarities among the handshake signals:

1. Handshake signals ACK and STB are input signals to the device and perform similar functions, although they are called by different names.
2. Handshake signals OBF and IBF are output signals from the device and perform similar functions (Buffer Full).

14.2 THE 8155: MULTIPURPOSE PROGRAMMABLE DEVICE

The 8155 is a multipurpose programmable device specially designed to be compatible with the 8085 microprocessor. The ALE, IO/M, RD, and WR signals from the 8085 can be connected directly to the device; this eliminates the need for external demultiplexing of the low-order bus AD₇ – AD₀ and generation of the control signals such as MEMR, MEMW, IOR, and IOW.

The 8155 includes 256 bytes of R/W memory, three I/O ports, and a timer. Figure 14.5 shows the pin configuration and block diagram of the 8155 programmable device. The programmable I/O sections of this device are illustrated in the following sections:

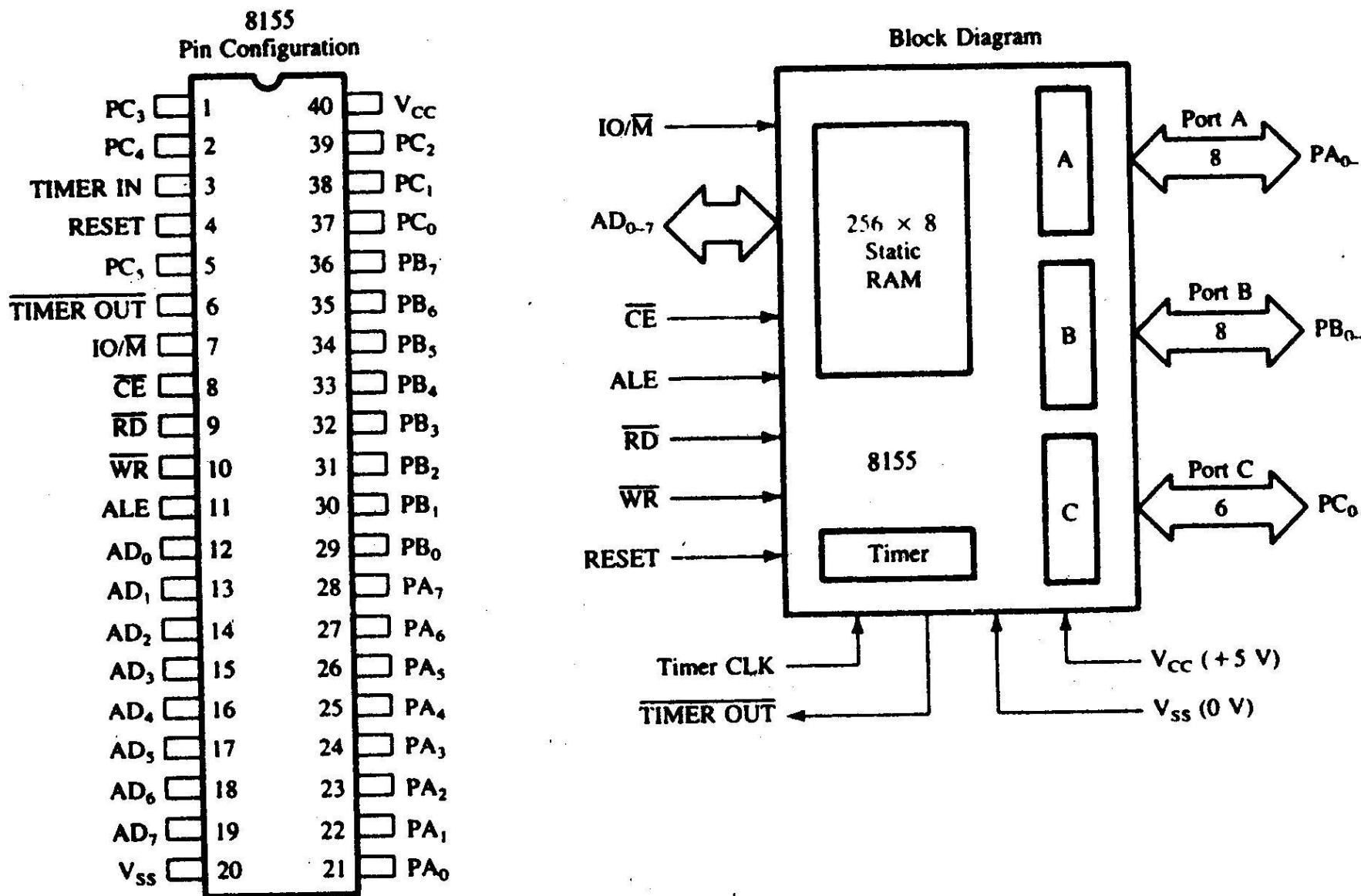


FIGURE 14.5
8155 Pin Configuration and Block Diagram

Source: Intel Corporation, *Embedded Microprocessors* (Santa Clara, Calif.: Author, 1994), pp. 1-31.

14.2 The 8155 Programmable I/O Ports and Timer

The 8155 is a device with two sections: the first is a 256 bytes R/W memory, and the second is a programmable I/O. Functionally, these two sections can be viewed as two independent chips.

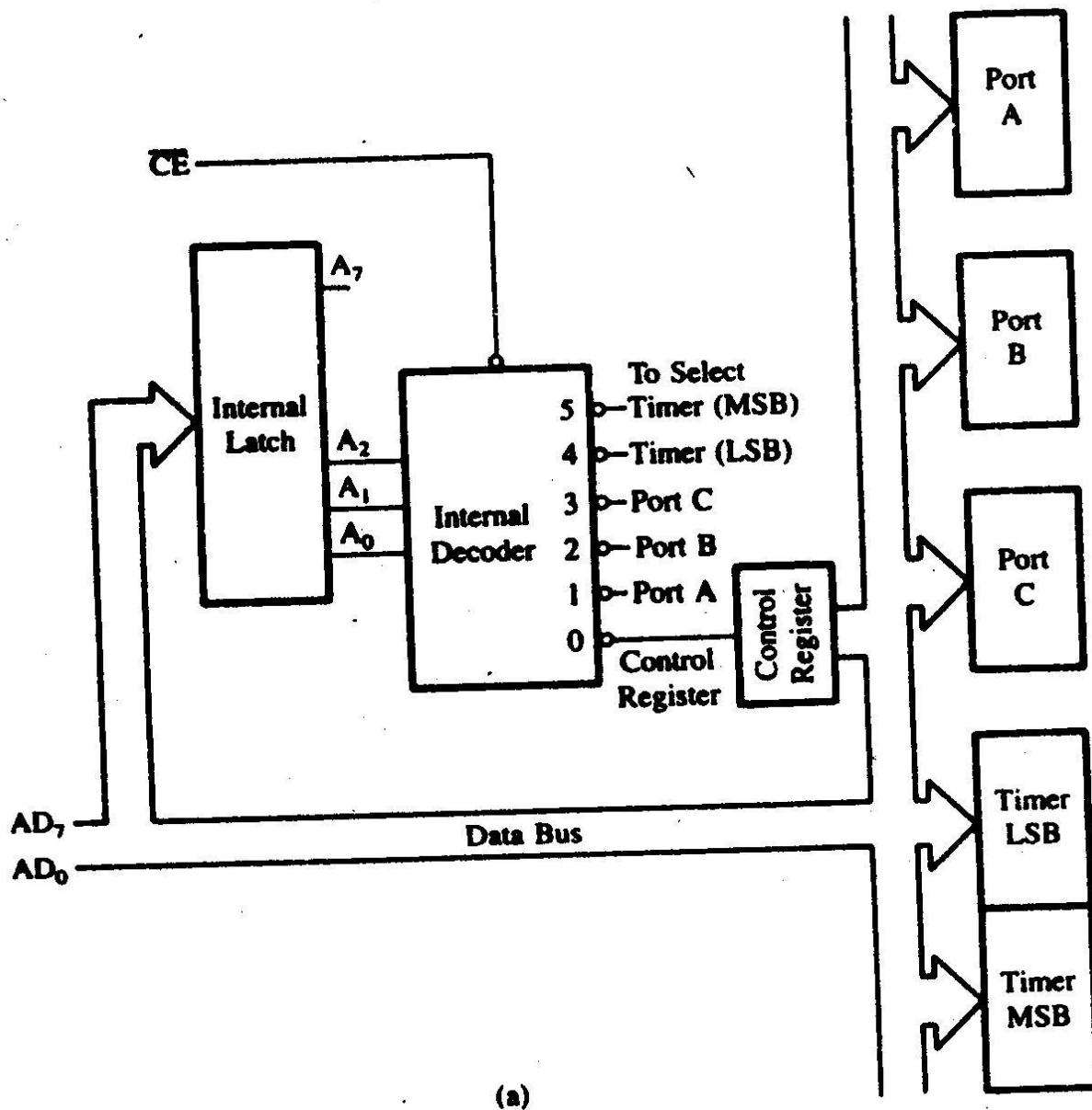
The I/O section includes two 8-bit parallel I/O ports (A and B), one 6-bit port (C), and a timer (Figure 14.5). All the ports can be configured simply as input/output ports. Ports A and B also can be programmed in the handshake mode, each port using three signals from port C. The timer is a 14-bit down-counter and has four modes.

THE 8155 I/O PORTS

The I/O section of the 8155 includes a control register, three I/O ports, and two registers for the timer (Figure 14.6).

To communicate with the peripherals through the 8155 the following steps are necessary:

1. Determine the address (port numbers of the registers and I/Os) based on the Chip Enable logic and address lines AD_0 , AD_1 , and AD_2 .



	A_2	A_1	A_0	Ports
Control/Status Register	0	0	0	Port A
Port A	0	0	1	Port B
Port B	0	1	0	Port C
Port C	0	1	1	LSB Timer
LSB Timer	1	0	0	MSB Timer
MSB Timer	1	0	1	

(b)

FIGURE 14.6
Expanded Block Diagram of the 8155 (a) and Its I/O Address: Selection (b)

2. Write control word in the control register to specify I/O functions of the ports and the timer characteristics.
3. Write I/O instructions to port addresses to communicate with peripherals.
4. Read the status register, if necessary, to verify the status of the I/O ports and the timer. In simple applications, this step is not necessary.

CHIP ENABLE LOGIC AND PORT ADDRESSES

Address lines $AD_2 - AD_0$ also shown as $A_2 - A_0$ after internal demultiplexing, select one of the registers, as shown in Figure 14.6(b). Address lines $A_3 - A_7$ are don't care lines; however, the logic levels on the corresponding high-order lines, $A_{11} - A_{15}$, will be duplicated on lines $A_3 - A_7$.

Example 14.2: Determine the addresses of the control/status register, I/O ports, and timer registers in Figure 14.7.

Solution: To select the chip, the output line of the 8205 (3-to-8) decoder (Figure 14.7) should go low:

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}
0	0	1	0	0

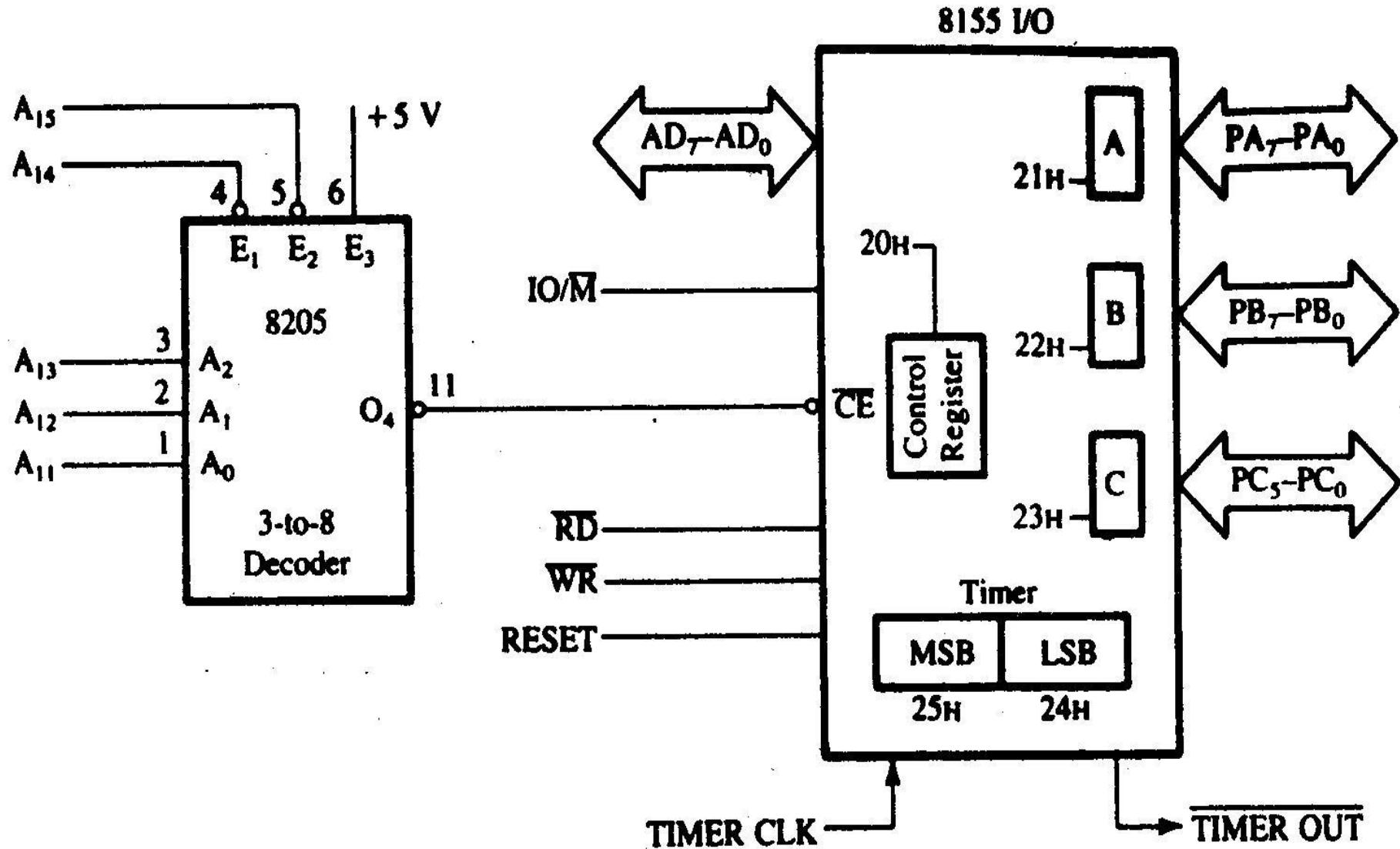


FIGURE 14.7

Interfacing 8155 I/O Ports (Schematic from the SDK-85 System)

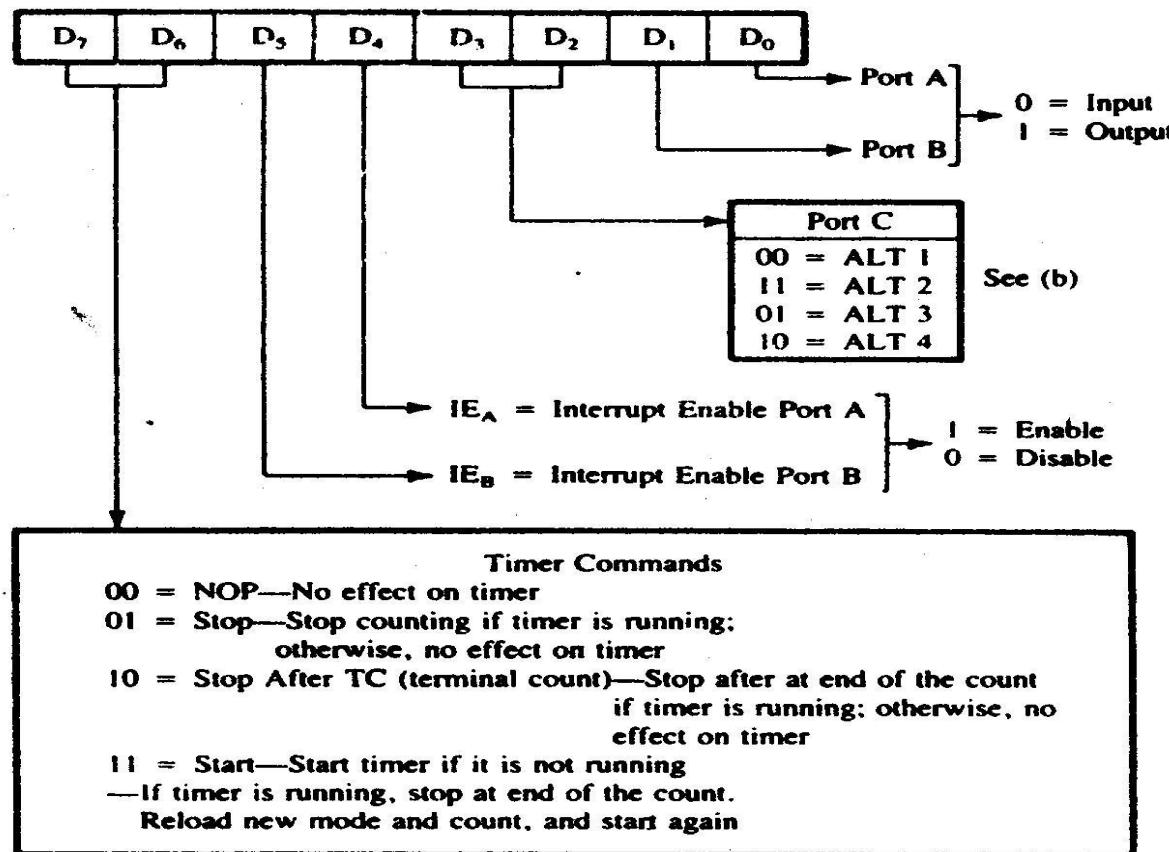
By combining five high-order address lines with three low-order address lines (A₂ – A₀), the port numbers in Figure 14.7 will range from 20H to 25H shown below:

A ₁₅	A ₁₄	A ₁₃	A ₁₂	A ₁₁	AD ₂	AD ₁	AD ₀ = Address	Ports
0	0	1	0	0	0	0	0	= 20H – CR or SR
					0	0	1	= 21H – Port A
					0	1	0	= 22H – Port B
					0	1	1	= 23H – Port C
					1	0	0	= 24H – Timer(LSB)
A ₇	A ₆	A ₅	A ₄	A ₃	1	0	1	= 25H – Timer(MSB)

N.B. For I/O-mapped I/O (or port-addressed I/O or peripheral I/O) the low-order address and high-order address bus carry the same information.

CONTROL WORD

The I/O ports and the timer can be configured by writing a control word in the control register. The control register bits are defined as shown in Figure 14.8.



(a)

Table: ALT 1–ALT 4: Port C Bit Assignments, Defined by Bits D₃ and D₂ in the Control Register

ALT	D ₃	D ₂	PC ₃	PC ₄	PC ₃	PC ₂	PC ₁	PC ₀
ALT 1	0	0	I	I	I	I	I	I
ALT 2	1	1	O	O	O	O	O	O
ALT 3	0	1	O	O	O	STB _A	BF _A	INTR _A
ALT 4	1	0	STB _B	BF _B	INTR _B	STB _A	BF _A	INTR _A

I = Input, STB = Strobe, INTR = Interrupt Request

O = Output, BF = Buffer Full, Subscript A = Port A

B = Port B

(b)

FIGURE 14.8

Control Word Definition in the 8155 (a) and Table of Port C Bit Assignments (b)

Bit D₂ and D₃ determine the functions of port C; their combination specifies one of the four alternatives, from simple I/O to interrupt I/O, as shown in Figure 14.8(b). Bits D₄ and D₅ are used only in the interrupt mode to enable or disable internal flip-flops of the 8155. These bits do not have any effect on the Internal Enable (INTE) flip-flop of the MPU.

14.2.2 Illustration: Interfacing Seven Segment LED Output Ports Using the 8155

PROBLEM STATEMENT

1. Design two seven-segment-LED displays using ports A and B of the 8155.
2. Write initialization instructions and display data bytes at each port.

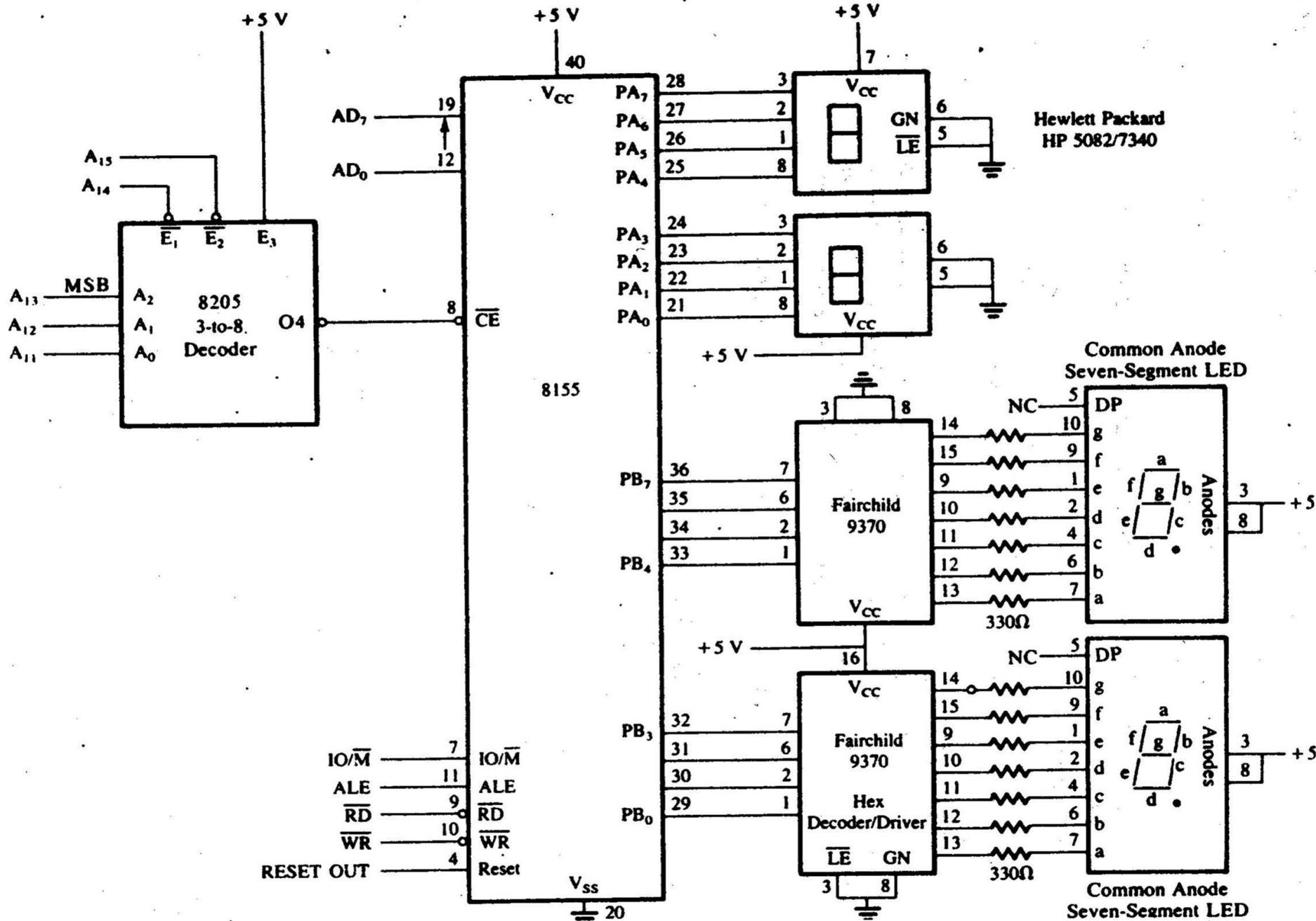
HARDWARE DESCRIPTION

The decode logic is the same as that used in the previous discussion; therefore, the port addresses are as follows:

Control Register = 20H

Port A = 21H

Port B = 22H



CONTROL WORD

To configure ports A and B as outputs, the control word is as follows:

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	=	03H
0	0	0	0	0	0	1	1		

PROGRAM

```
MVI A, 03H  
OUT 20H  
MVI A, BYTE1  
OUT 21H  
MVI A,BYTE2  
OUT 22H  
HLT
```

14.2.3 The 8155 Timer

The timer section of the 8155 has two 8-bit registers; 14 bits are used for the counter, two bits for the timer mode, and it requires a clock as an input. This 14-bit down counter provides output in four different modes, as described in Figure 14.(b).

The timer can be stopped either in the midst of counting or at the end of a count (applicable to Modes 1 and 3).

14.2.4 Illustration: Designing a Square-Wave Generator Using the 8155 Timer

PROBLEM STATEMENT

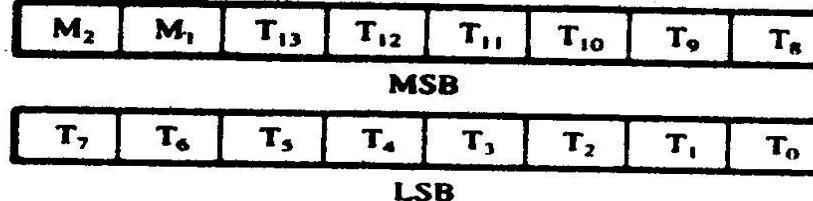
Design a square-wave generator with a pulse width of $100 \mu s$ by using the 8155 timer. Set up timer in Mode 1 if the clock frequency is 3 MHz. Use the same decode logic and the port addresses as in Example 14.2 (Figure 14.7).

PROBLEM ANALYSIS

Timer Count: The pulse width required is $100 \mu s$; therefore, the count should be calculated for the period of $200 \mu s$. The timer output stays high for only half the count.

202

Loading Format



(a)

Modes

Mode 0: In this mode, the timer output remains high for half the count and goes low for the remaining count, thus providing a single square wave. The pulse width is determined by the count and the clock frequency.

Mode 1: In this mode, the initial timer count is automatically reloaded at the end of each count, thus providing a continuous square wave.

Mode 2: In this mode, a single clock pulse is provided at the end of the count.

Mode 3: This is similar to Mode 2, except the initial count is reloaded to provide a continuous wave form.

$M_2\ M_1$

0 0

Single Square Wave Cycle



0 1

Square Wave



1 0

Single Pulse Upon Terminal Count



1 1

Pulse Every Terminal Count



N = Count in Timer

(b)

FIGURE 14.10

Timer Loading Format (a) and Modes (b)

Clock Period = $1/f = 1/3 \times 10^6 = 330$ ns

Timer Count = Pulse Period / Clock Period = $200 \times 10^{-6} / 330 \times 10^{-9} = 606$

Count = 025EH

The port addresses for the timer registers are

Timer LSB = 24H

Timer MSB = 25H

The least significant byte, 5EH (of the count 025EH), should be loaded in the timer register with address 24H. The most significant byte is determined as follows:

M_2	M_1	T_{13}	T_{12}	T_{11}	T_{10}	T_9	T_8	
0	1	0	0	0	0	1	0	= 42H

Therefore, 42H should be loaded in the timer register with the address 25H.

CONTROL WORD

Ports A and B are used as output port; therefore, the control word is:

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	=	C3H
1	1	0	0	0	0	1	1		

Initialization Instructions

```
MVI A, 5EH  
OUT 24H  
MVI A,42H  
OUT 25H  
MVI A, C3H  
OUT 20H  
HLT
```

14.2.5 The 8155 I/O Ports in Handshake Mode

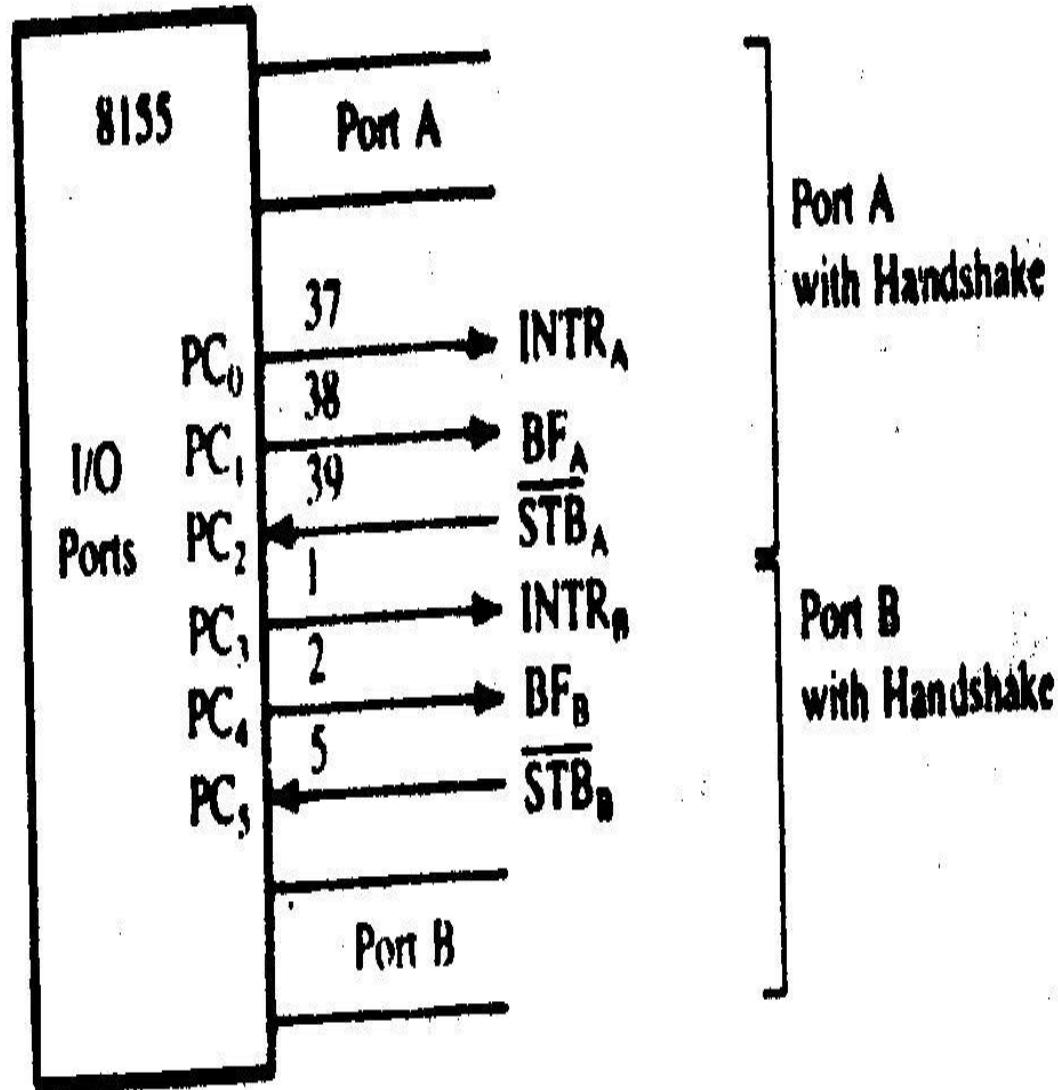
In the handshake mode, data transfer occurs between the MPU and peripherals using control signals called handshake signals. Two I/O ports of the 8155, A and B, can be configured in the handshake mode; each uses three signals from port C as control signals (Figure 14.11). Another alternative (ALT3 in the table in Figure 14.8) available in the 8155 is to configure port A in the handshake mode with three control signals from port C, configure port B as simple I/O, and configure the remaining three bits from port C as outputs.

CONTROL SIGNALS IN HANDSHAKE MODE

When both ports A and B are configured in the handshake mode, port A uses the lower three signals of port C (PC_0 , PC_1 , and PC_2), and port B uses the upper three signals (PC_3 , PC_4 , and PC_5), as shown in Figure 14.11. The function of these signals are:

- **STB (Strobe Input):** The low signal on this pin informs the 8155 that data are strobed into the input port.
- **BF (Buffer Full):** The high signal on this pin indicates the presence of a data byte in the port.

FIGURE 14.11
8155 with Handshake Mode



- **INTR (Interrupt Request)**: This signal interrupts the processor to read an existing data from its (8155) input port or write a new data to its output port.
- **INTE (Interrupt Enable)** : This is an internal flip-flop used to enable or disable the interrupt capability of the 8155. The interrupts for port A and B are controlled by bits D₄ and D₅ respectively, in the control register.

These control signals can be used to implement either interrupt I/O or status check I/O.

INPUT

Figure 14.12 (a) shows the sequence of events and timing in data input to the 8155.

- 1.
- 2.
- 3.
- 4.

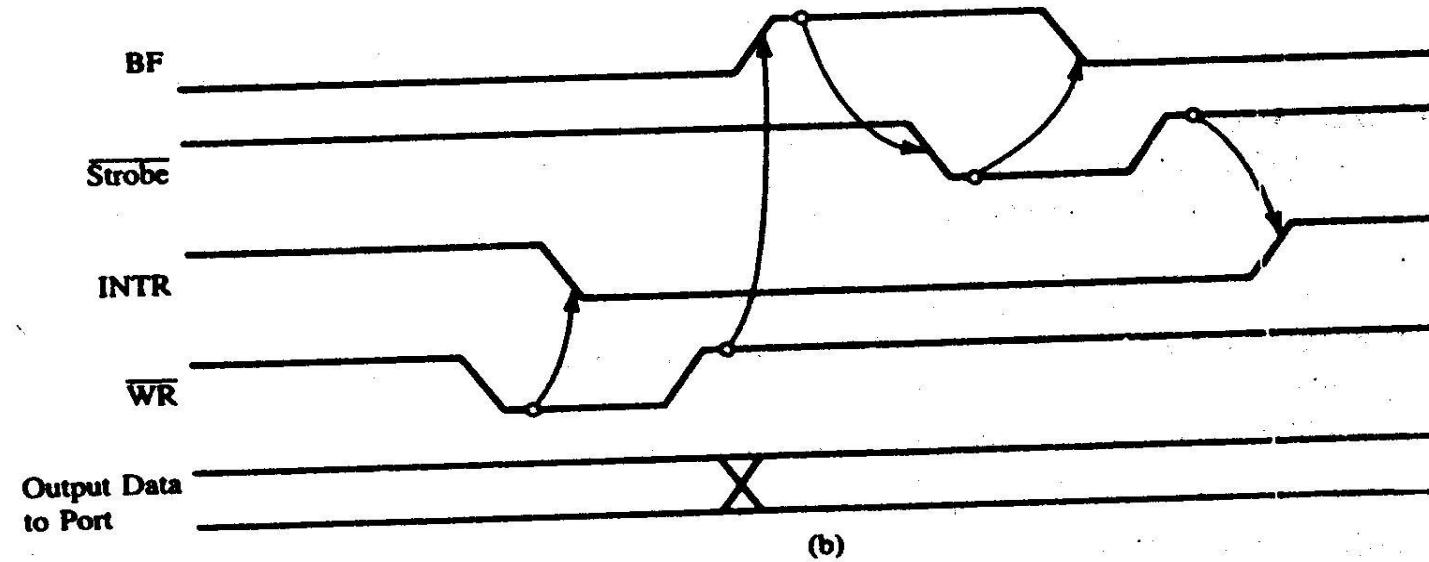
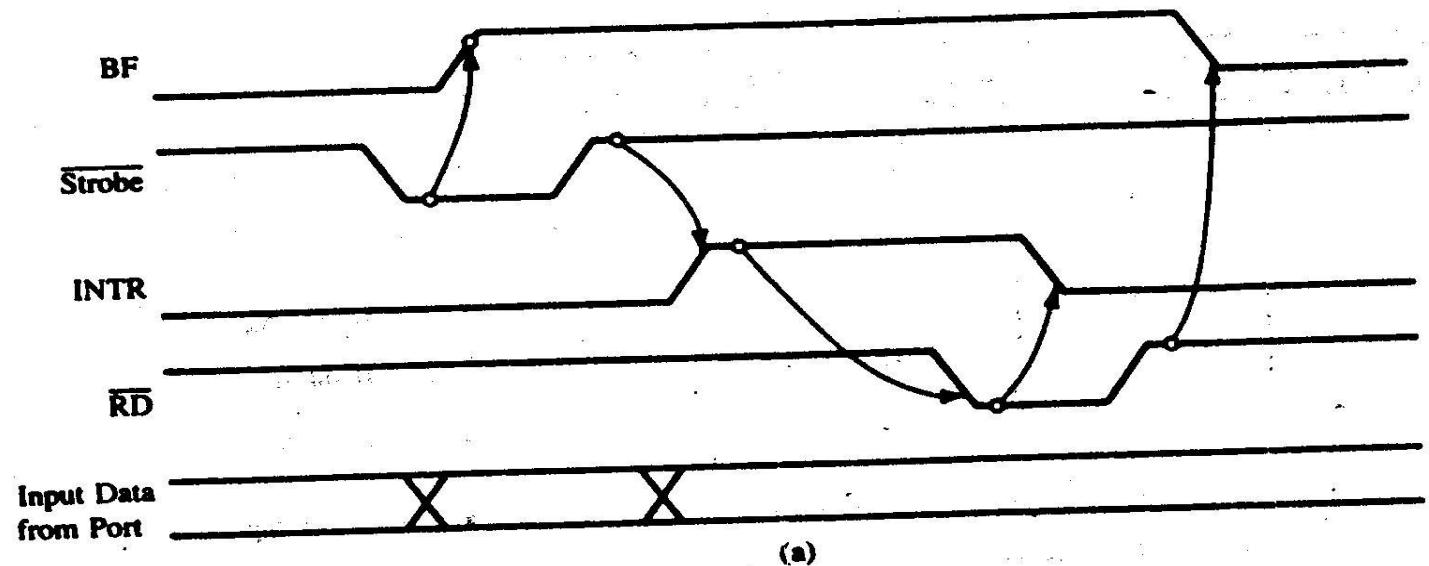


FIGURE 14.12
Timing Waveforms of the 8155 I/O Ports with Handshake: Input Mode (a) and Output Mode (b)

— 14.14 — Microprocessors (Santa Clara, Calif.: Author, 1994), pp. 1-43.

OUTPUT

Figure 14.12 (b) shows the sequence of events and timing in data output to the 8155.

- 1.
- 2.
- 3.
- 4.

STATUS WORD

The MPU can read the status register to check of the timer. The control register and the status register have the same port address; they are differentiated only by RD and WR signals. The status register bits are defined in Figure 14.13.

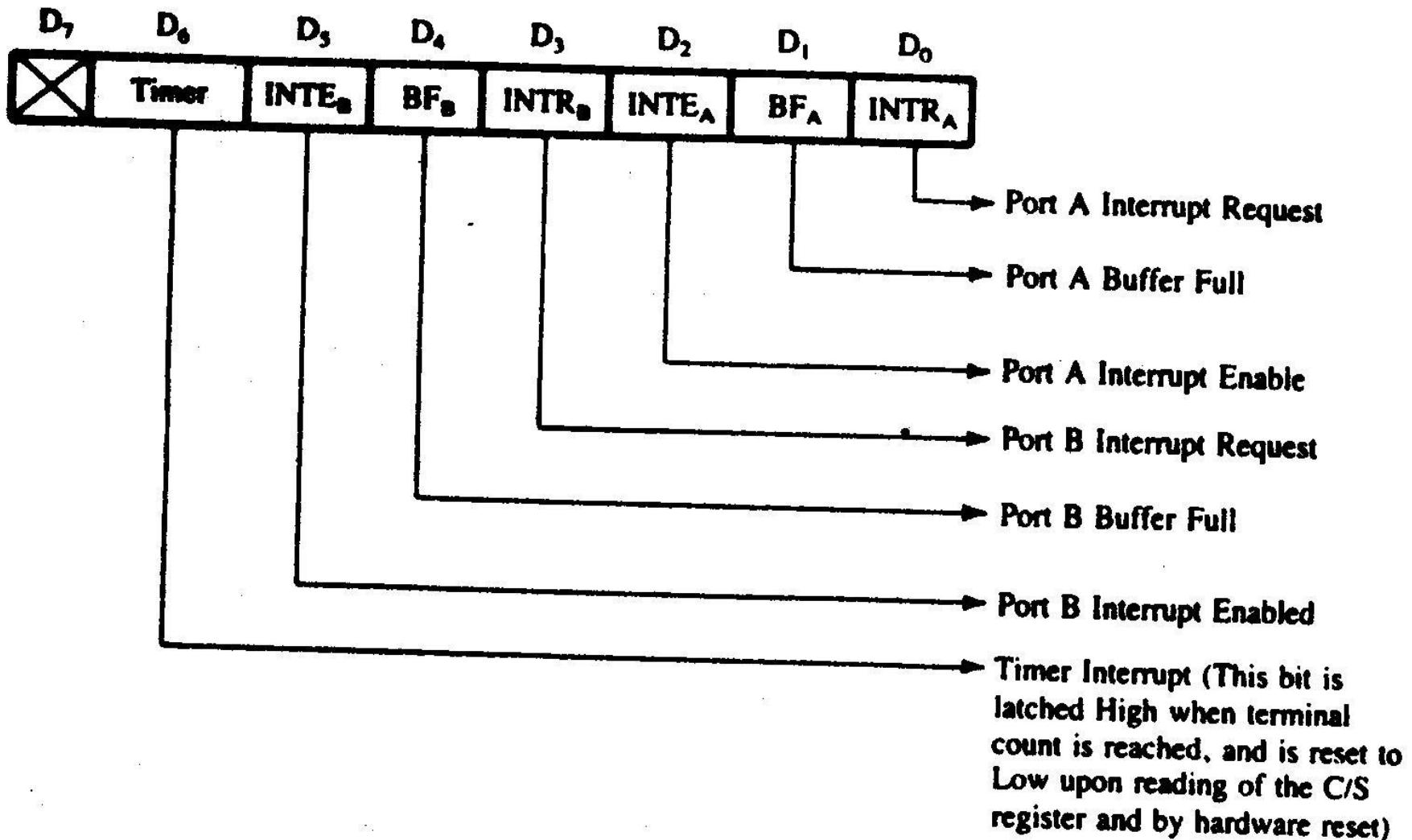


FIGURE 14.13
Status Word Definition

Source: Intel Corporation, *MCS-80/85 Family User's Manual* (Santa Clara, Calif.: Author, 1979), pp. 6-20.

14.2.6 Illustration: Interfacing I/O Ports in Handshake Mode using the 8155

PROBLEM STATEMENT

Design an interfacing circuit using the 8155 read and display form an A/D converter to meet the following requirements:

1. Set up port A in the handshake mode to read data from A/D converter.
2. Set up port B as an output port to display data at seven-segment LEDs.
3. Use line PC_3 from port C to initiate a conversion.

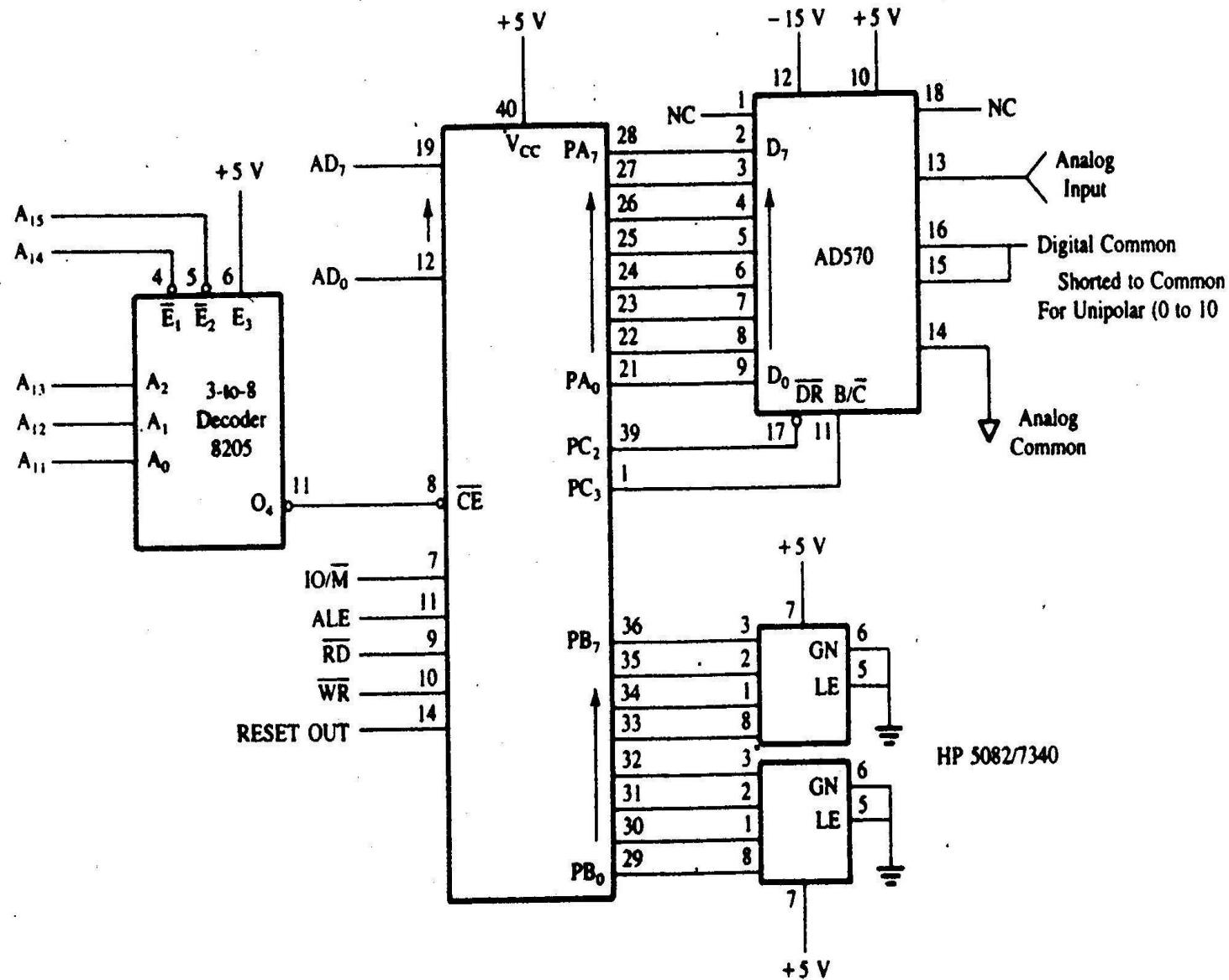


FIGURE 14.14

Interfacing the A/D Converter AD570 in the Handshake Mode

PROBLEM ANALYSIS

The circuit shows that the INTR signal (bit PC_0) is not being used. This suggests that port A is configured for status check and not for interrupt I/O. Therefore, the control word:

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	=	06H
0	0	0	0	0	1	1	0		

8155 Timer

To start the timer, the control word:

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	=	C6H
1	1	0	0	0	1	1	0		

To stop the timer, the control word:

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	=	46H
0	1	0	0	0	1	1	0		

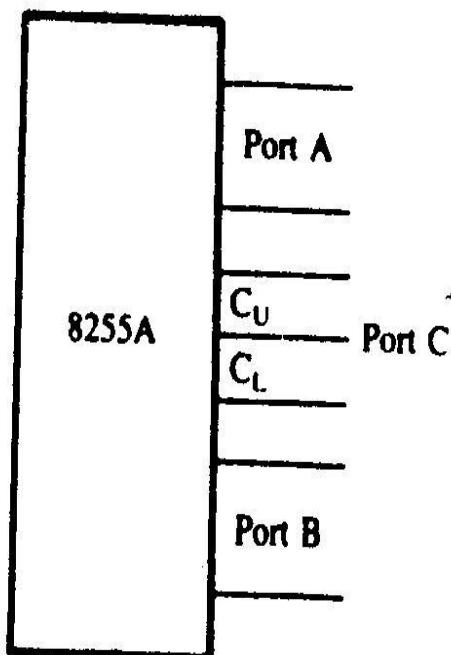
Chapter-15

General Purpose Programmable Peripheral Devices

15.1 THE 8255A PROGRAMMABLE PERIPHERAL INTERFACE

The 8255A is a widely used, programmable, parallel I/O device. It can be programmed to transfer data under various conditions, from simple I/O to interrupt I/O. It is flexible, versatile, and economical (when multiple I/O ports are required), but somewhat complex. It is an important general-purpose I/O device that can be used with almost any microprocessor.

The 8255A has 24 I/O pins that can be grouped primarily in two 8-bit parallel ports: A and B, with the remaining 8-bits as port C. The eight bits of port C can be used as individual bits or grouped in two 4-bit ports: C_{UPPER} (C_U) and C_{LOWER} (C_L), as in Figure 15.1(a). The functions of these ports are defined by writing control word in the control register.



(a)

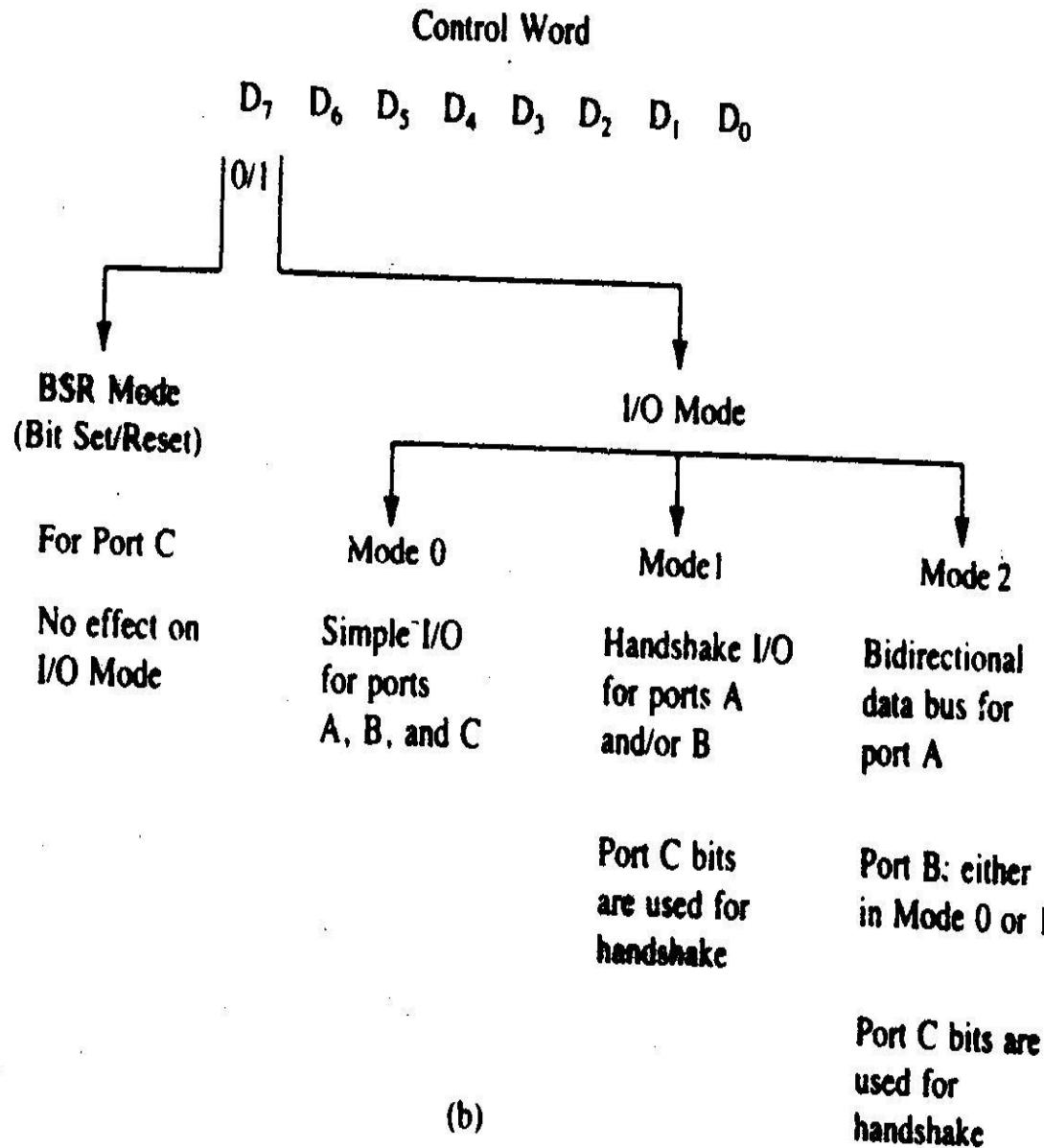


FIGURE 15.1

8255A I/O Ports (a) and Their Modes (b)

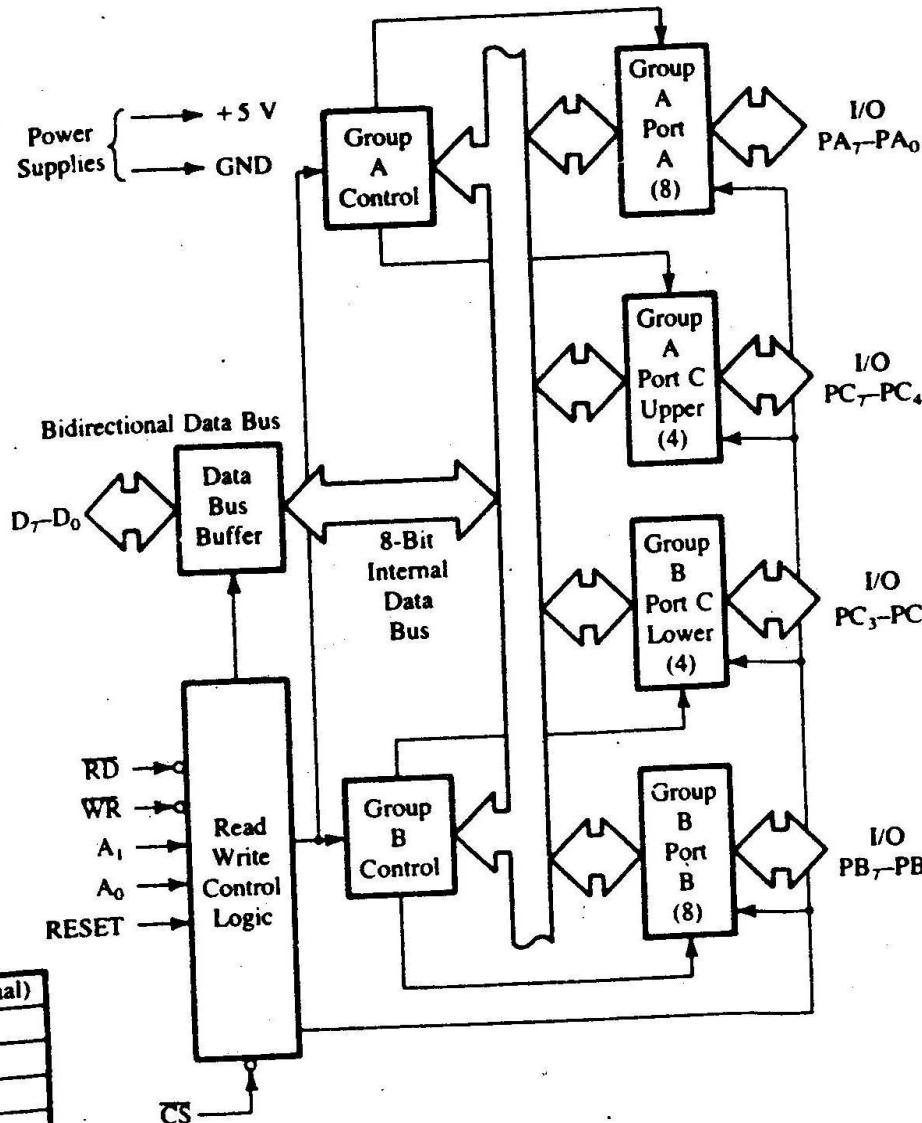
Figure 15.1(b) shows all the functions of the 8255A, classified according to two modes: the Bit Set/Reset (BSR) mode and the I/O mode. The BSR mode is used to set or reset the bits in port C. The I/O mode is further divided into three modes: Mode 0, Mode 1, and Mode 2.

- **Mode 0**: all ports function as simple I/O ports.
- **Mode 1**: handshake mode; ports A and/or B use bits from port C as handshake signals.
- **Mode 2**: port A for bidirectional data transfer using handshake signals from port C, and port B can be either Mode 0 or Mode 1.

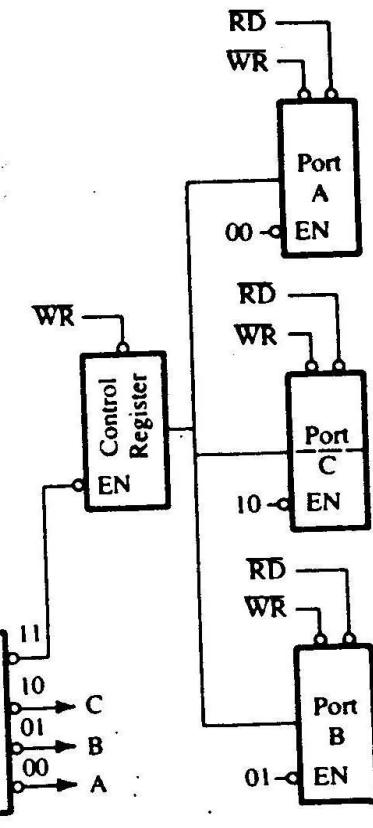
15.1.1 Block Diagram of the 8255A

The block diagram in Figure 15.2(a) shows two 8-bit ports (A and B), two 4-bit ports (C_U and C_L).

Pin Configuration	
PA ₃	1
PA ₂	2
PA ₁	3
PA ₀	4
RD	5
CS	6
GND	7
A ₁	8
A ₀	9
PC ₇	10 8255A
PC ₆	11
PC ₅	12
PC ₄	13
PC ₀	14
PC ₁	15
PC ₂	16
PC ₃	17
PB ₀	18
PB ₁	19
PB ₂	20
	21
PA ₄	40
PA ₅	39
PA ₆	38
PA ₇	37
WR	36
RESET	35
D ₀	34
D ₁	33
D ₂	32
D ₃	31
D ₄	30
D ₅	29
D ₆	28
D ₇	27
V _{CC}	26
PB ₃	25
PB ₆	24
PB ₅	23
PB ₄	22
PB ₃	21



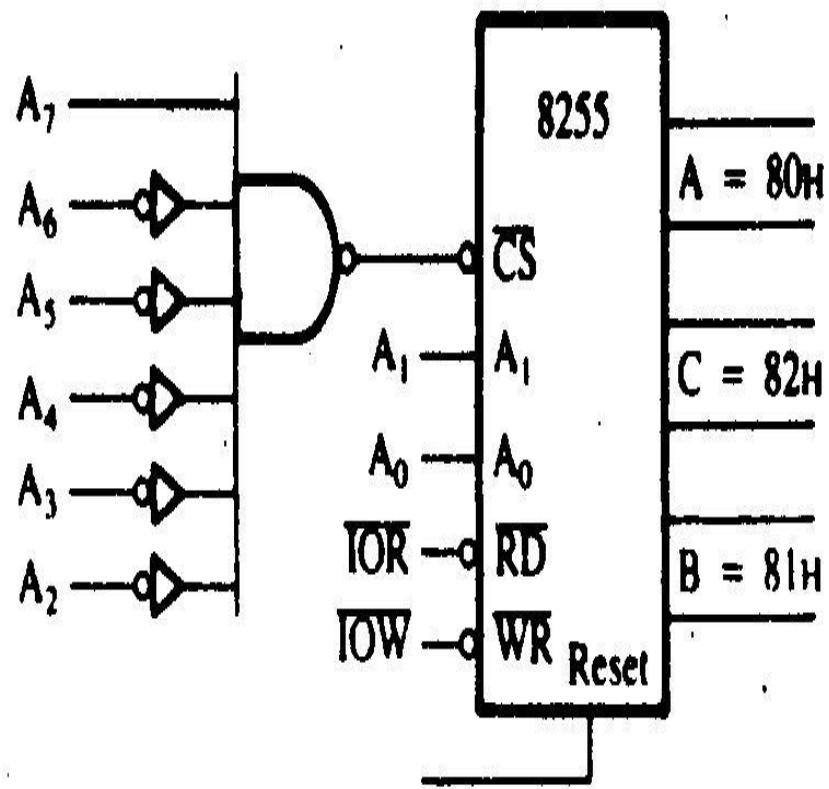
(a)



(b)

FIGURE 15.2

Detailed Version of the Control Logic and I/O Ports (b)



(a)

CS		Hex Address	Port
$A_7 A_6 A_5 A_4 A_3 A_2$	$A_1 A_0$	$= 80H$	A
1 0 0 0 0 0	0 0	$= 81H$	B
	0 1	$= 82H$	C
	1 0	$= 83H$	Control Register
	1 1		

(b)

FIGURE 15.3

8255A Chip Select Logic (a) and I/O Port Addresses (b)

CONTROL WORD

The bit D₇ specifies either the I/O function or the Bit Set/Reset function as classified in Figure 15.1(b). Figure 15.4 shows the control word format for I/O Mode.

To communicate with the peripherals through the 8255A, three steps are necessary through the 8255A, three steps are necessary:

1. Determine the addresses of ports A, B, and C and of the control register according to the Chip Select logic and A₀ and A₁.
2. Write a control word in the control register.
3. Write I/O instructions to communicate through ports A, B, and C.

15.1.2 Mode 0: Simple Input or Output

In this mode, ports A and B are used as two simple 8-bit I/O ports and port C as two 4-bit ports. Each port (or half-port, in case of C) can be programmed to function as simply an input port or an output port. The input/output features in Mode 0 are as follows:

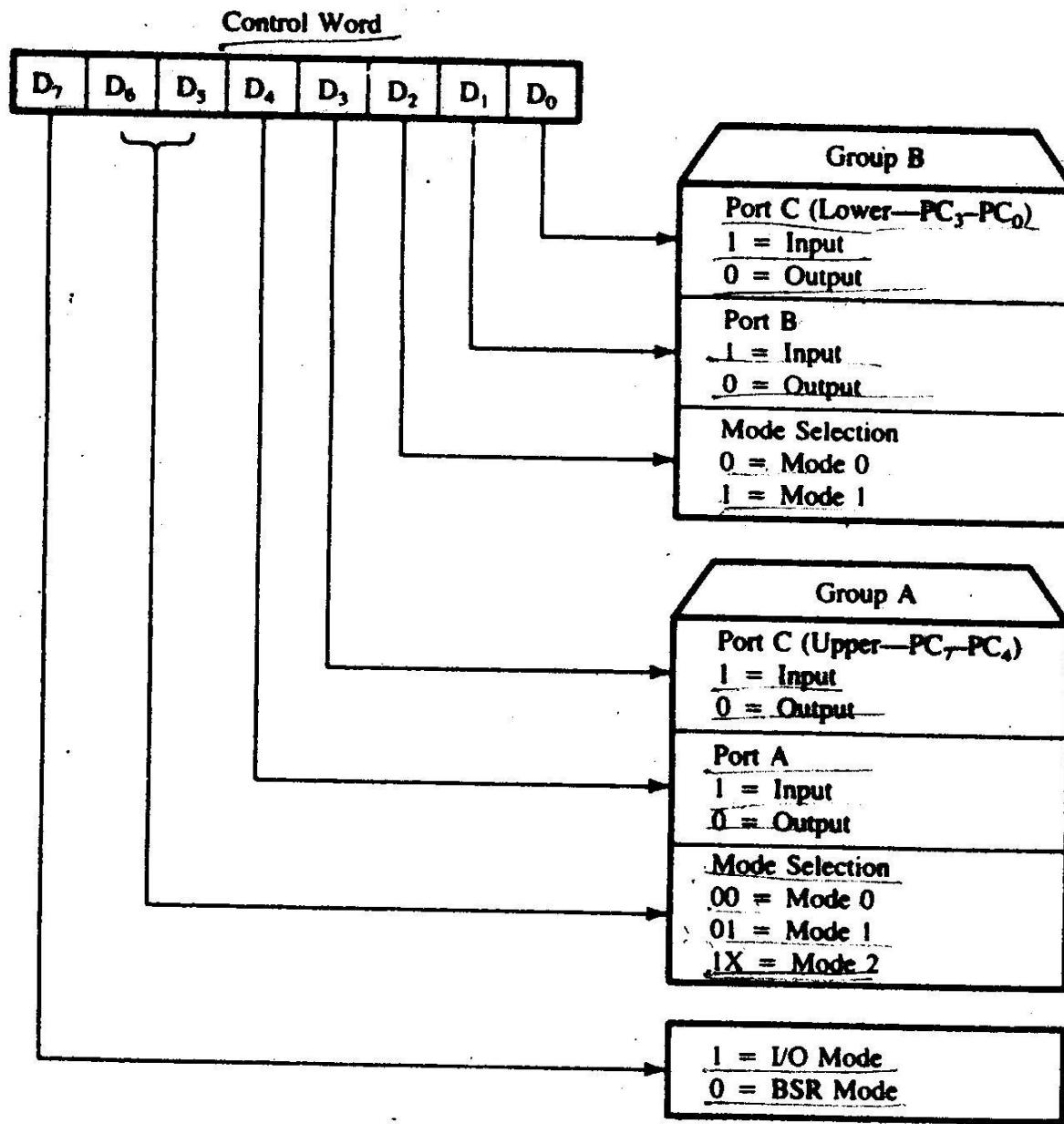


FIGURE 15.4
8255A Control Word Format for I/O Mode

SOURCE: Adapted from Intel Corporation, *Peripheral Components* (Santa Clara, Calif.: Author, 1993), p. 3-104.

1. Outputs are latched.
2. Inputs are not latched.
3. Ports do not have handshake or interrupt capability.

Example 15.1:

1. Identify the port addresses in Figure 15.5.
2. Identify the Mode 0 control word to configure port A and port C_U as output ports and port B and port C_L as input ports.
3. Write a program to read the DIP switches and display the reading from port B port A and from port C_L at port C_U.

Solution:

1. Port Addresses

Port A = 8000H (A₁=0, A₀=0)

Port B = 8001H (A₁=0, A₀=1)

Port C = 8002H (A₁=1, A₀=0)

Control Register = 8003H (A₁=1, A₀=1)

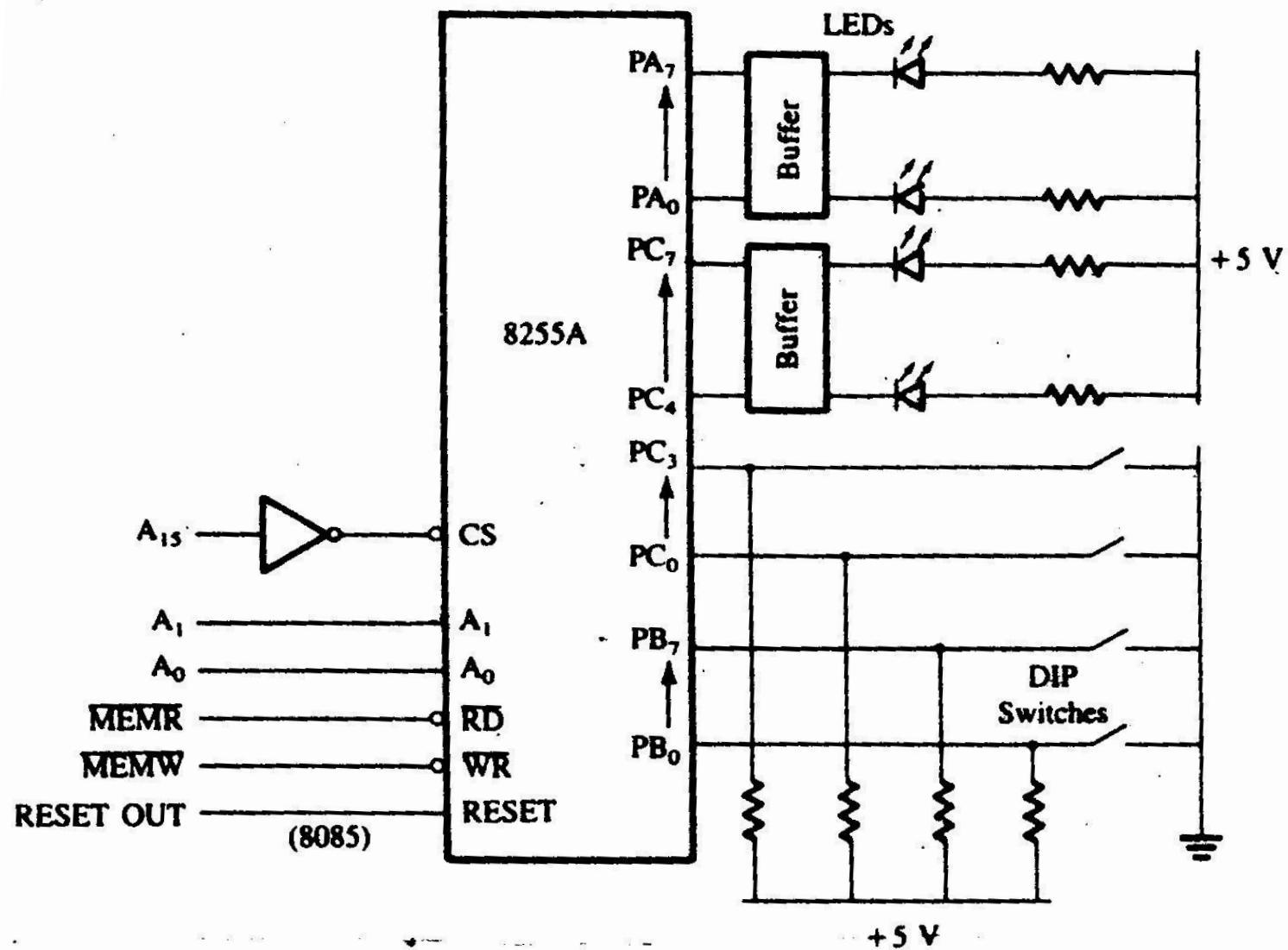


FIGURE 15.5
Interfacing 8255A I/O Ports in Mode 0

2. Control Word

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
1	0	0	0	0	0	1	1

= 83H

3. Program

```
MVI A,83H  
STA 8003H  
LDA 8001H  
STA 8000H  
LDA 8002H  
ANI 0FH  
RLC  
RLC  
RLC  
RLC  
STA 8002H  
HLT
```

15.1.3 BSR (Bit Set/Reset) Mode

The BSR mode is concerned only with the eight bits of port C, which can be set or reset by writing an appropriate control word in the control register. A control word with bit $D_7=0$ is recognized as a BSR control word, and it does not alter any previously transmitted control word with bit $D_7=1$; thus the I/O operations of ports A and B are not affected by a BSR control word. In the BSR mode, individual bits of port C can be used for applications such as an on/off switch.

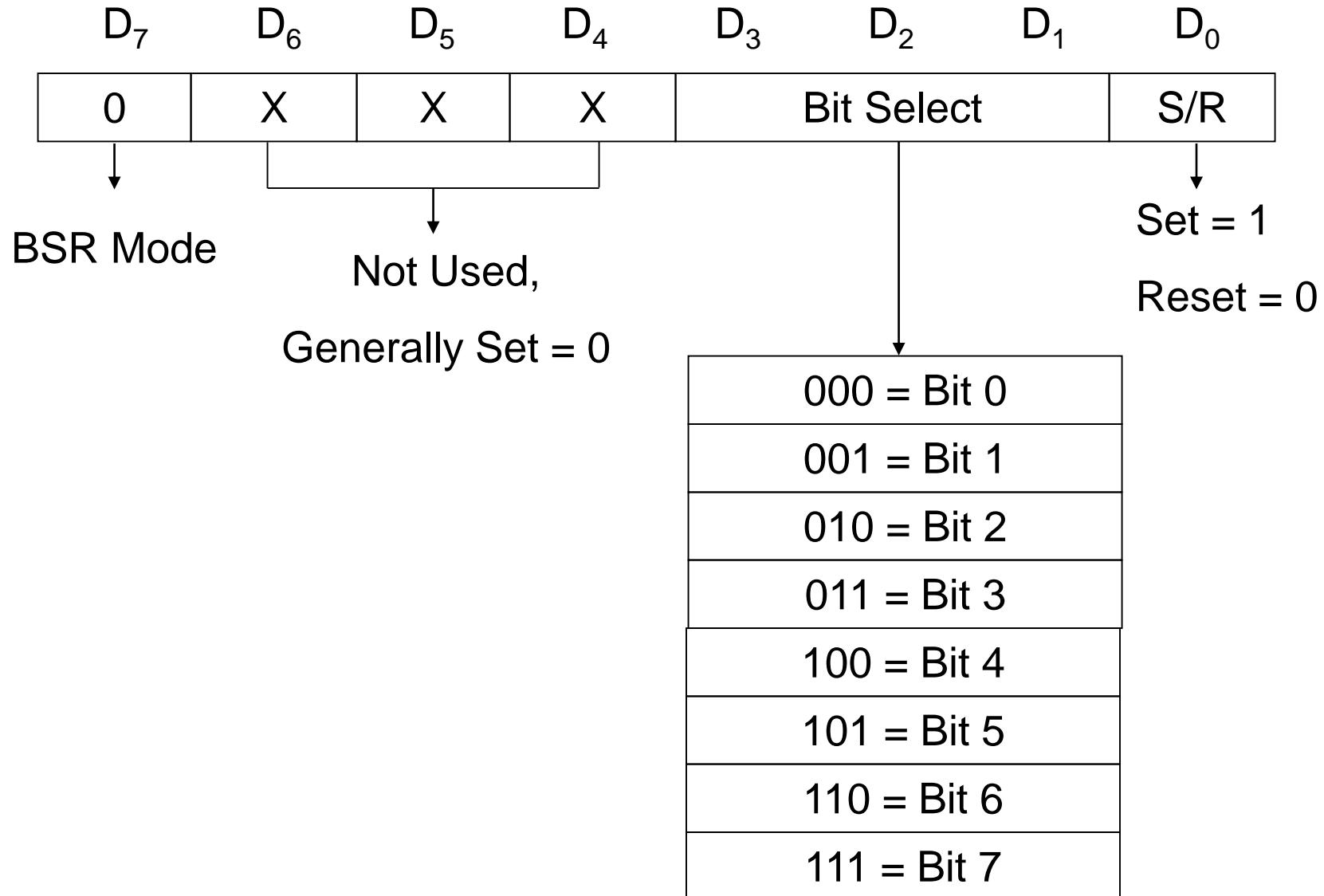
BSR CONTROL WORD

This control word, when written in the control register, sets or resets one bit at a time, as specified in Figure 15.6.

Example 15.2 :

Write BSR control word subroutine to set bits PC_7 and PC_3 and reset them after 10 ms. Use the schematic in Figure 15.3 and assume that a delay subroutine is available.

Figure 15.6 : 8255A Control Word Format in the BSR Mode



BSR Control Words

	D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	=	
To set bit PC ₇	= 0	0	0	0	1	1	1	1	=	0FH
To reset bit PC ₇	= 0	0	0	0	1	1	1	0	=	0EH
To set bit PC ₃	= 0	0	0	0	0	1	1	1	=	07H
To reset bit PC ₃	= 0	0	0	0	0	1	1	0	=	06H

PORT ADDRESS

Control register address = 83H; refer to Figure 15.3(b).

SUBROUTINE

BSR: MVI A, 0FH
OUT 83H
MVI A,07H
OUT 83H
CALL DELAY
MVI A,06H
OUT 83H
MVI A,0EH
OUT 83H
RET

15.4 THE 8254 (8253) PROGRAMMABLE INTERVAL TIMER

The 8254 programmable interval timer/counter is functionally similar to the software designed counters and timers described in chapter 8. It generates accurate time delay and can be used for applications such as a real-time clock, an event counter, a square-wave generator, and a complex waveform generator.

The 8254 includes three identical 16-bit counters that can operate independently in any one of the six modes. It is packaged in a 24-pin DIP and requires a single +5v power supply. To operate a counter, a 16-bit count is loaded in its register and, on command, begins to decrement the count until it reaches 0. At the end of the count, it generates a pulse that can be used to interrupt the MPU. The counter can count in binary or BCD. In addition, count can be read by the MPU while counter is decrementing.

15.4.1 Block Diagram of the 8254

Figure 15.23 is the block diagram of the 8254; it includes three counters (0, 1, 1 and 2), a data bus buffer, Read/Write control logic, and a control register. Each counter has two input signals-Clock (CLK) and GATE-and output signal-OUT.

Pin Configuration

D ₇	1	24	V _{CC}
D ₆	2	23	WR
D ₅	3	22	RD
D ₄	4	21	CS
D ₃	5	20	A ₁
D ₂	6	8254	A ₀
D ₁	7	19	CLK 2
D ₀	8	18	OUT 2
CLK 0	9	17	GATE 2
OUT 0	10	16	CLK 1
GATE 0	11	15	GATE 1
GND	12	14	OUT 1
		13	OUT 1

Pin Names

D ₇ -D ₀	Data Bus (8 Bit)
CLK N	Counter Clock Inputs
GATE N	Counter Gate Inputs
OUT N	Counter Outputs
RD	Read Counter
WR	Write Command or Data
CS	Chip Select
A ₀ -A ₁	Counter Select
V _{CC}	+ 5 Volts
GND	Ground

Block Diagram

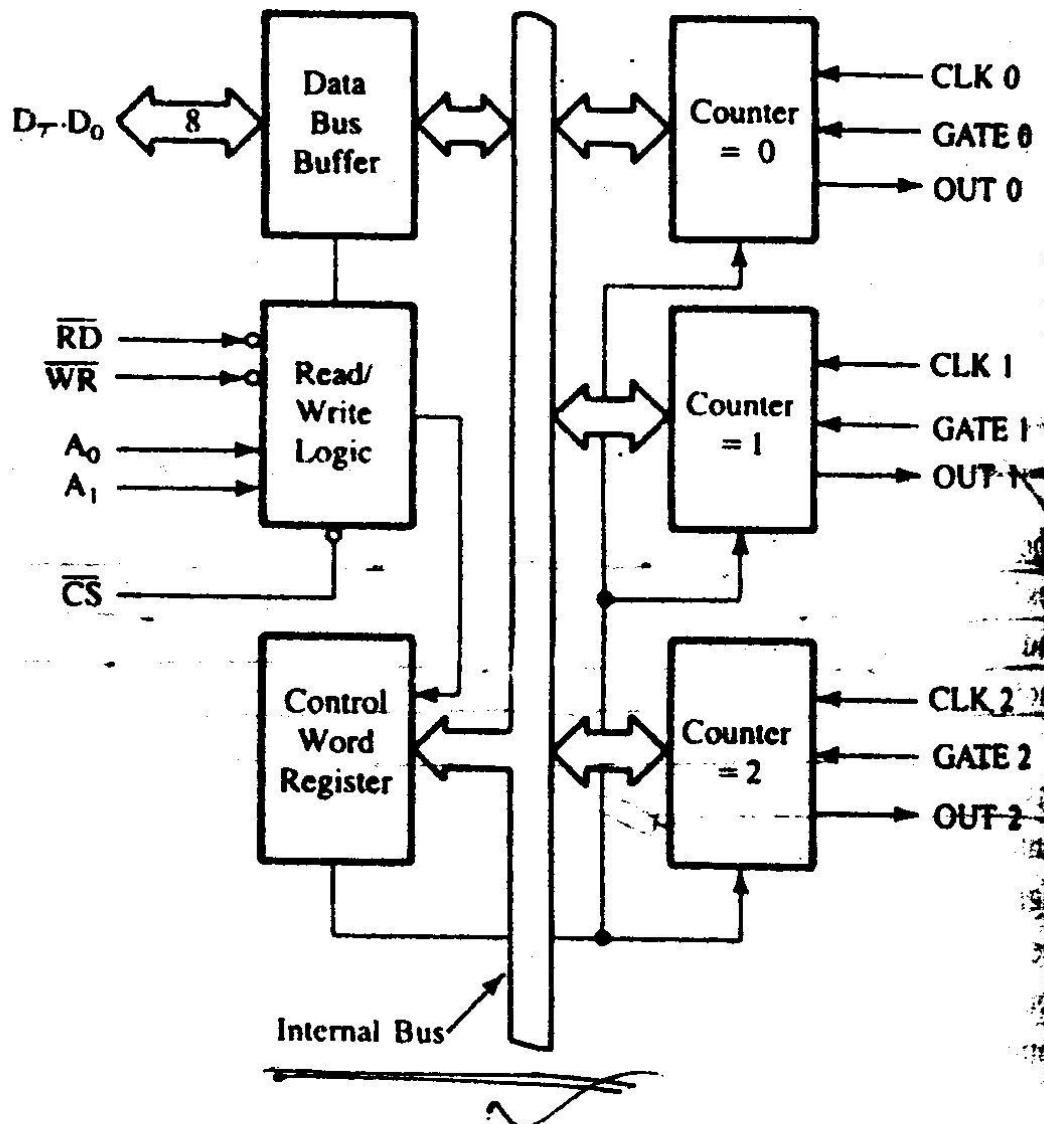


FIGURE 15.23

8254 Block Diagram

SOURCE: Intel Corporation, *Peripheral Components* (Santa Clara, Calif.: Author, 1993), p. 3-62.

CONTROL LOGIC

<u>A₀</u>	<u>A₁</u>	<u>Selection</u>
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Control Register

CONTROL WORD REGISTER

The control word format is shown in Figure 15.24.

MODE

8254 Control Word Format

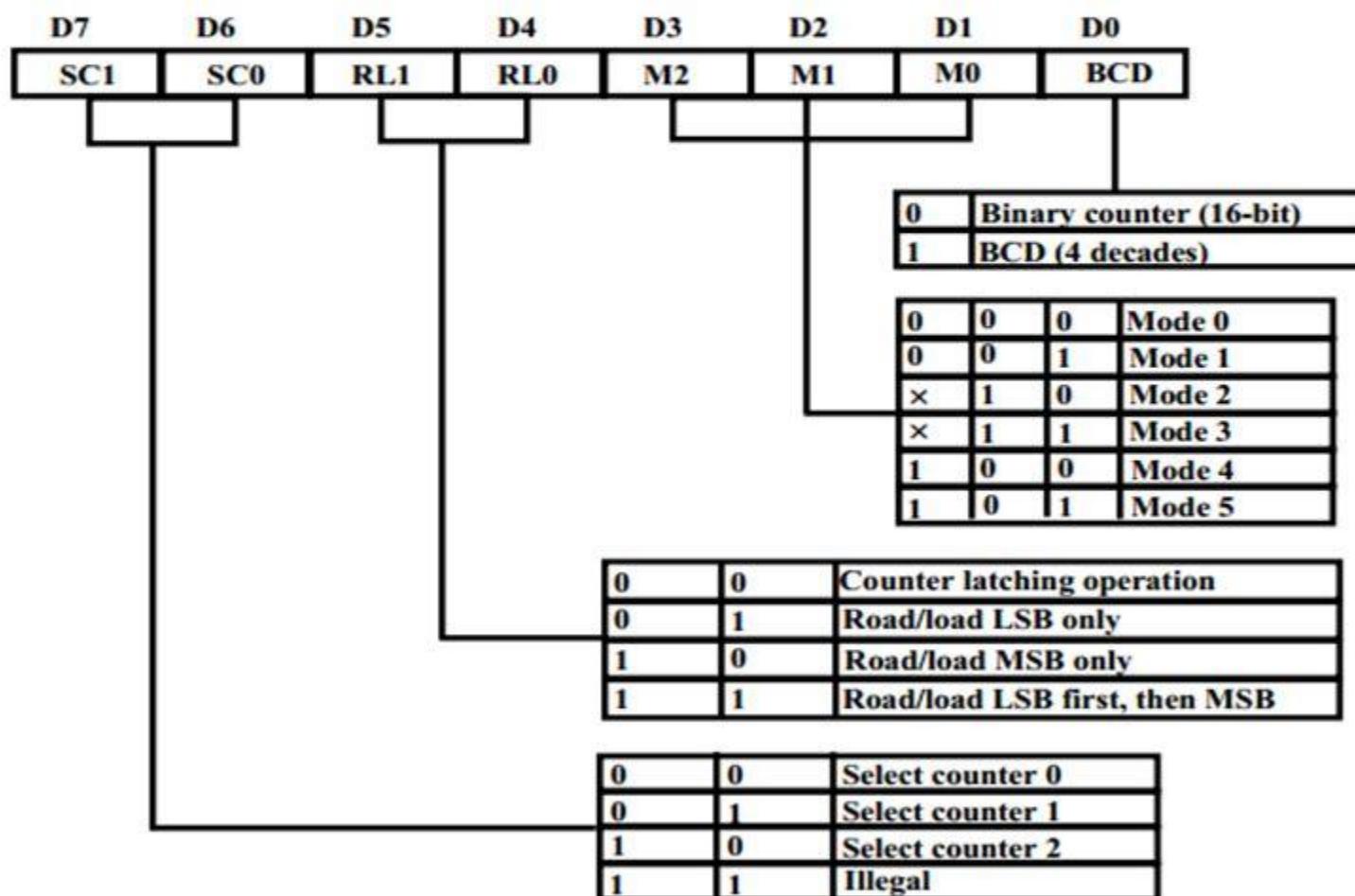


Fig. 14.9 Control Register