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:

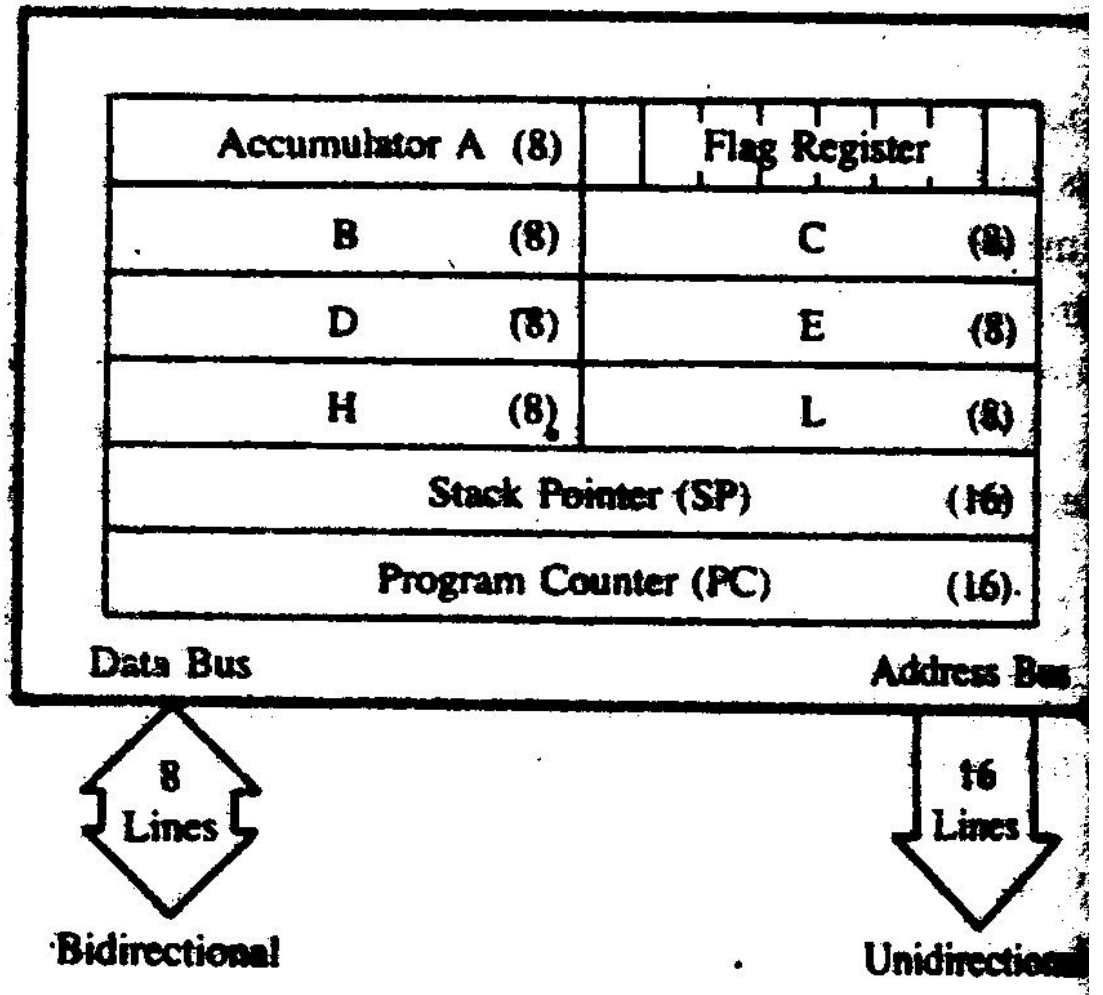
- Step 1: Identify the peripheral or the memory location (with its address)
- Step 2: Transfer data.
- Step 3: Provide timing or synchronization signals.

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.

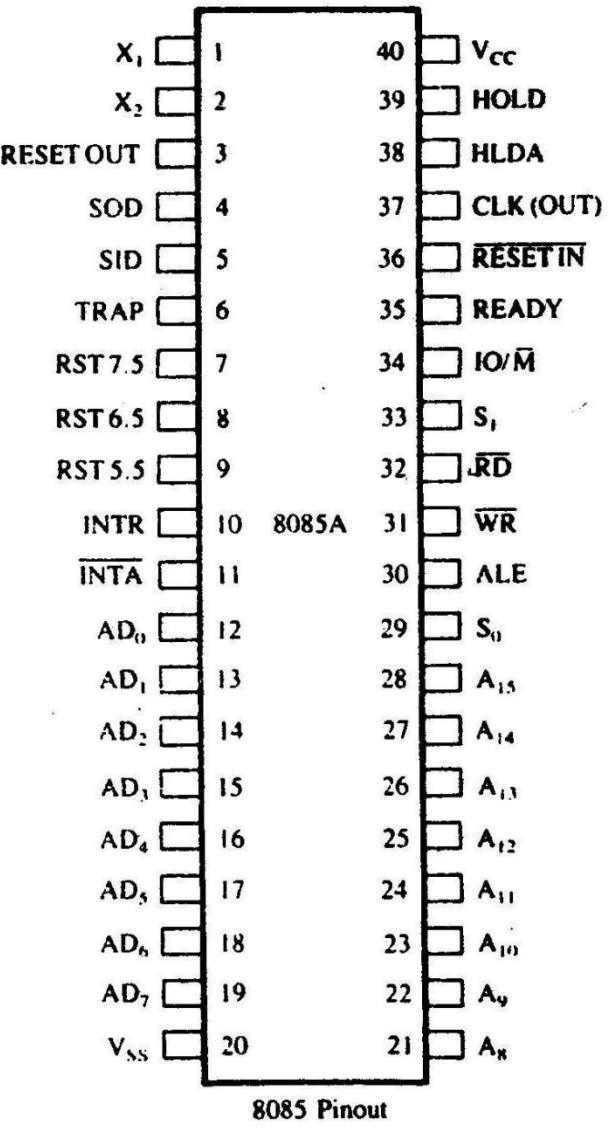
FIGURE 2.3
The 8085 Programmable Registers



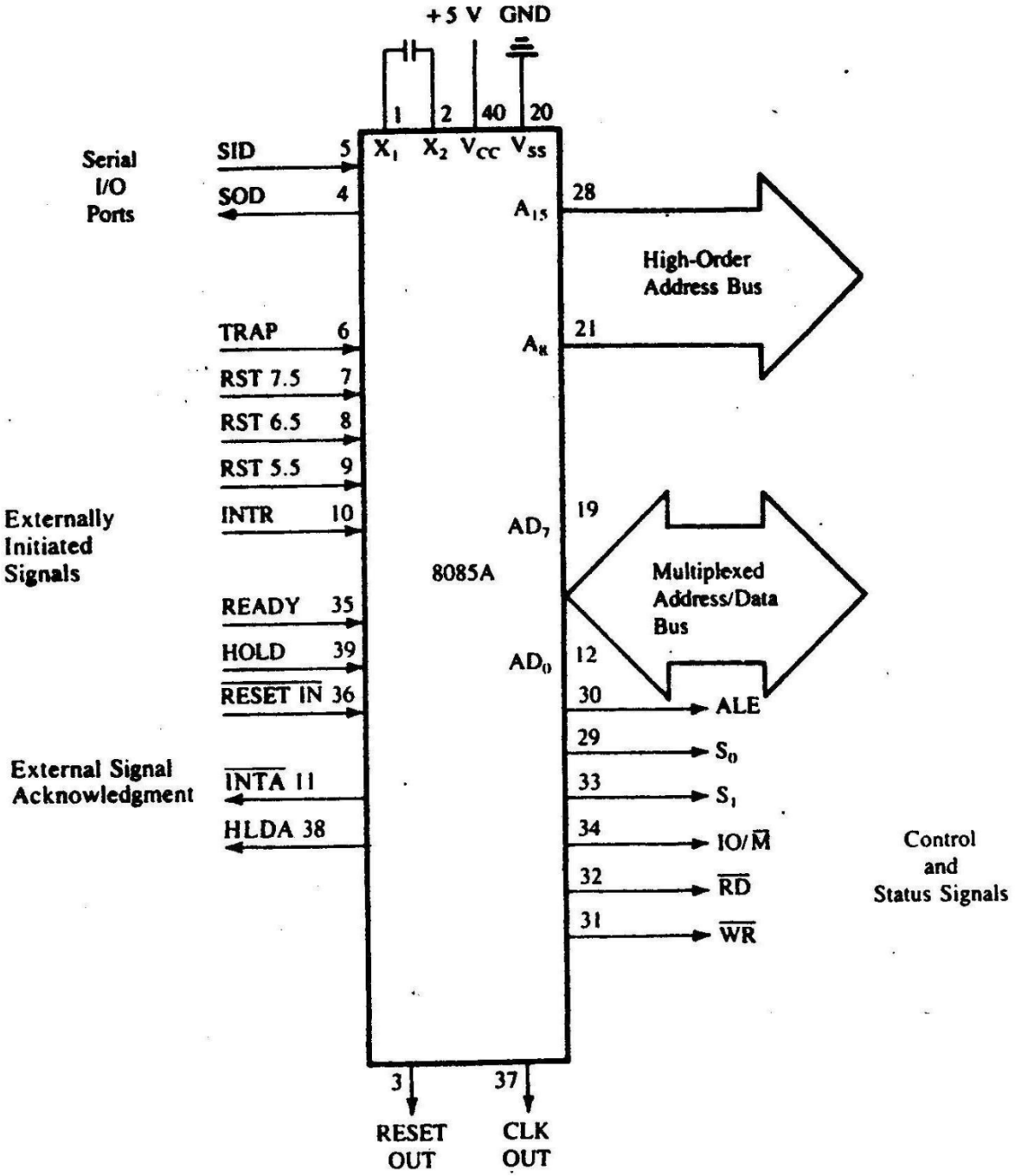
2.1.3 Peripheral or Externally Initiated Operations

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.


X1 1
X2 2
RESET OUT 3
SOD 4
SID 5
TRAP 6
RST 7.5 7
RST 6.5 8
RST 5.5 9
INTR 10
INTA 11
AD0 12
AD1 13
AD2 14
AD3 15
AD4 16
AD5 17
AD6 18
AD7 19
VSS 20
8085A
40 VCC
39 HOLD
38 HLDA
37 CLK (OUT)
36 RESET IN
35 READY
34 IO/M
33 S1
32 RD
31 WR
30 ALE
29 S0
28 A15
27 A14
26 A13
25 A12
24 A11
23 A10
22 A9
21 A8

8085 Pinout


+5 V
GND
1
2
40
20
X1
X2
VCC
VSS
Serial I/O Ports
SID 5
SOD 4
TRAP 6
RST 7.5 7
RST 6.5 8
RST 5.5 9
Externally Initiated Signals
INTR 10
READY 35
HOLD 39
RESET IN 36
External Signal Acknowledgment
INTA 11
HLDA 38
8085A
A15 28
A8 21
High-Order Address Bus
AD7 19
AD0 12
Multiplexed Address/Data Bus
30 ALE
29 S0
33 S1
34 IO/M
32 RD
31 WR
Control and Status Signals
3
37
RESET OUT
CLK OUT

Chapter-3

8085 Microprocessor Architecture and Memory Interfacing

3.1 THE 8085 MPU

Microprocessor Unit (MPU)

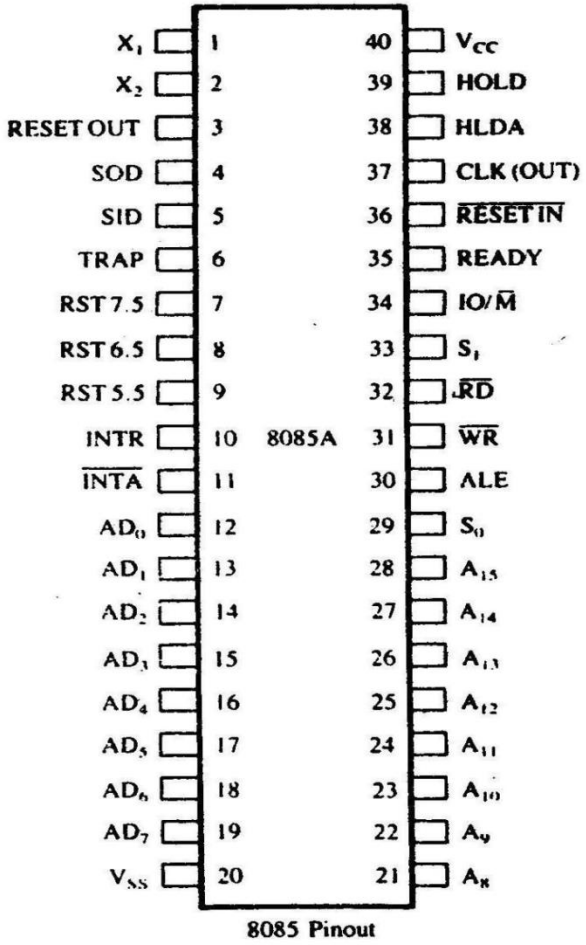
❑ The term microprocessor unit (MPU) define as a group of devices (as a unit) that can communicate with peripherals, provide timing signals, direct data flow, and perform computing tasks as specified by the instructions in memory.

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.



8085 Pinout

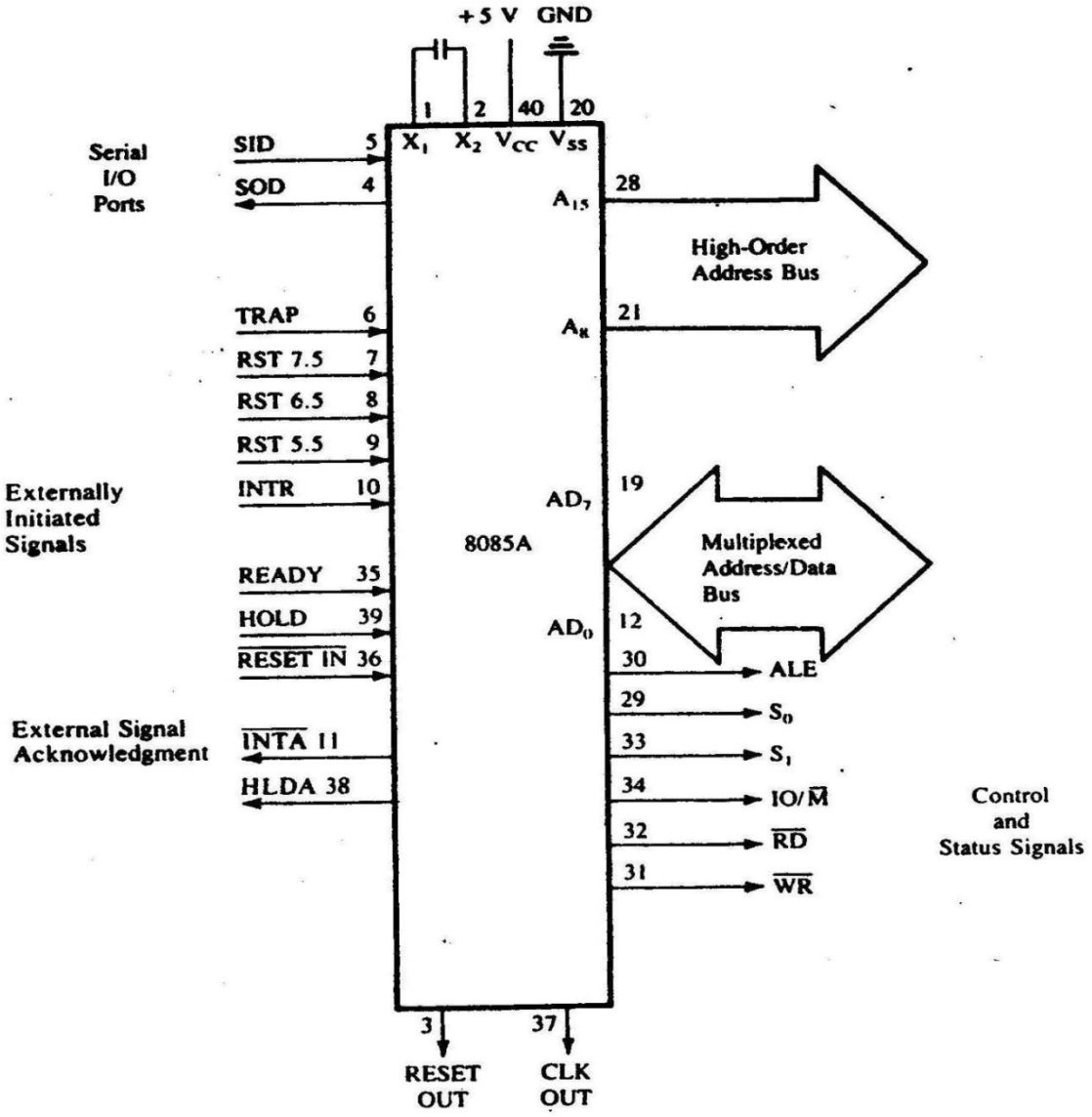


FIGURE 3.1
The 8085 Microprocessor Pinout and Signals

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

- A_{15} - A_8 are unidirectional and high order address bus.

MULTIPLEXED ADDRESS/DATA BUS

- AD_7 - AD_0 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: $\overline{RD}$ and $\overline{WR}$.
- Status Signals: IO/$\overline{M}$, S_1 , and S_0.
- Special Signal: ALE. Indicate the beginning of the operation.
- $\overline{RD}$: Low for read operation(active low).
- $\overline{WR}$: Low for write operation (active low).
- IO/$\overline{M}$: 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 AD_7 - AD_0 are address bits. This signal are used to latch the low-order address bus an generate a separate set of eight address lines; A_7 - A_0

EXTERNALLY INITIATED SIGNALS INCLUDING INTERRUPT

TABLE 3.2
8085 Interrupts and Externally Initiated Signals

- INTR (Input)
- $\overline{\text{INTA}}$ (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/$\overline{M}$ goes low.

Step 2: The control unit sends the control signal $\overline{RD}$ to enable the memory chip (Figure 3.2).

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

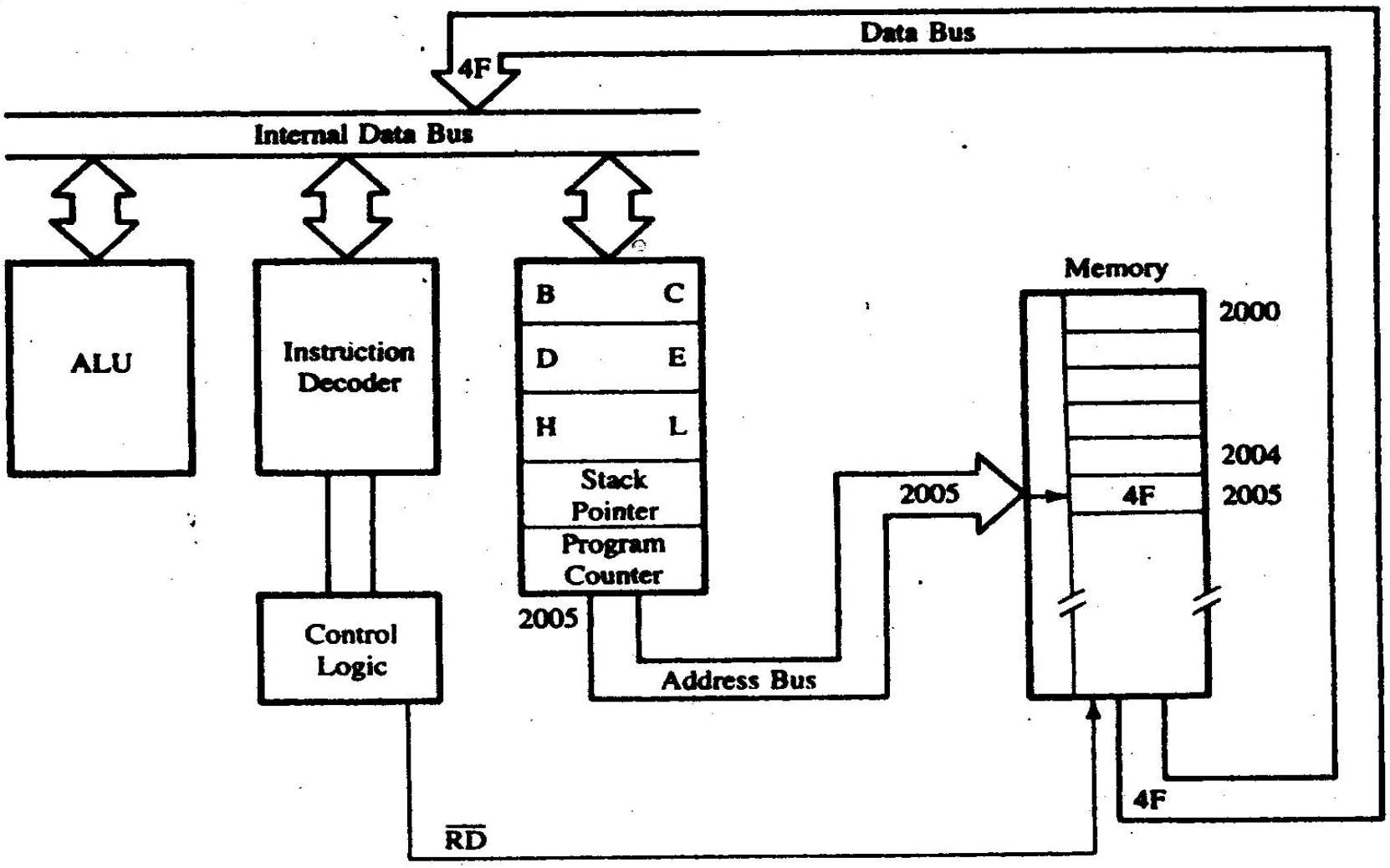


FIGURE 3.2
Data Flow from Memory to the MPU

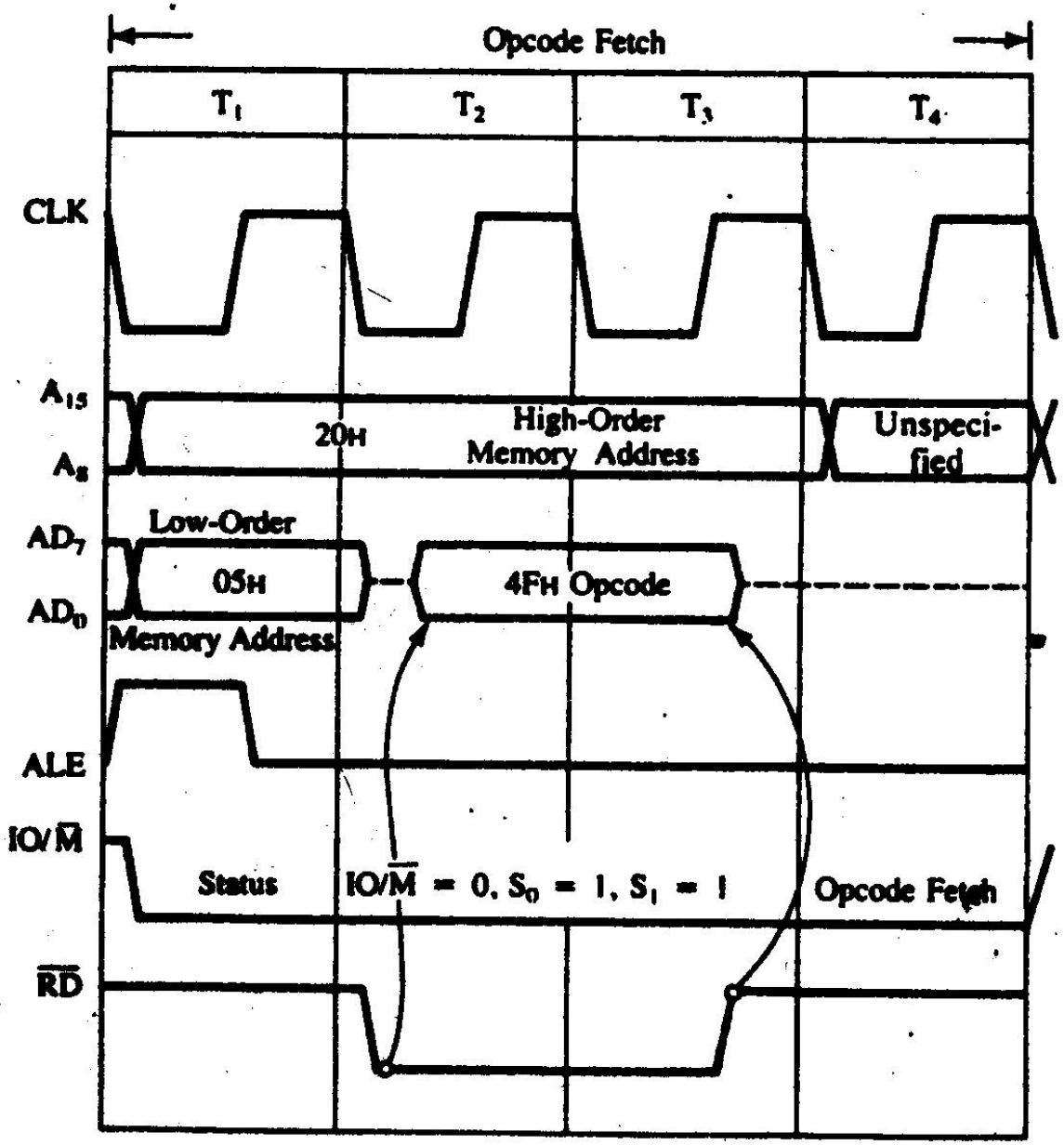


FIGURE 3.3
Timing: Transfer of Byte from Memory to MPU

The $\overline{RD}$ signal causes 4FH to be placed on bus AD_7 - AD_0 (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 AD_7-AD_0

Figure 3.3 shows that the address on the high-order bus (20H) remains on the bus for three clock periods (T_1 to T_3). 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.

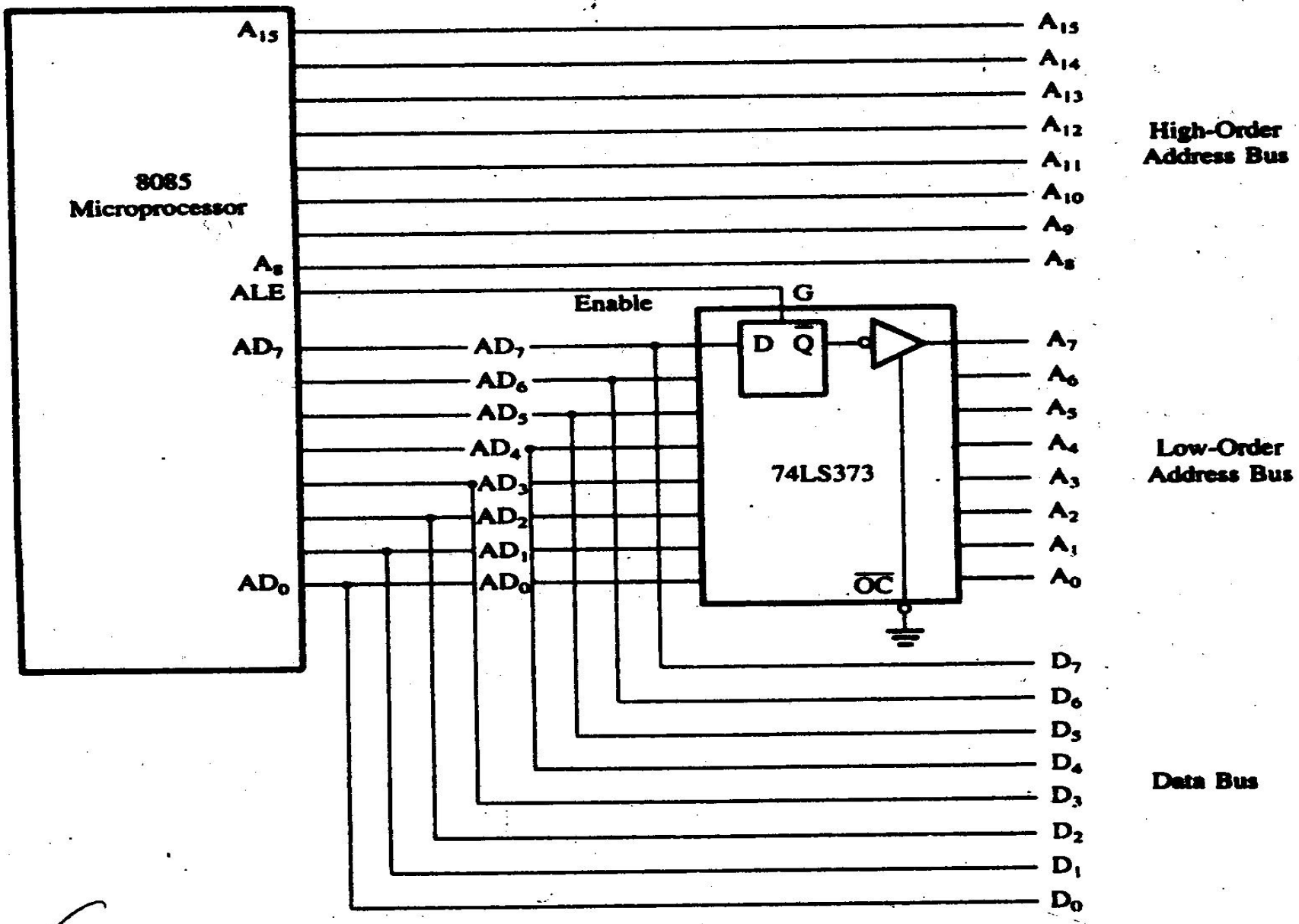


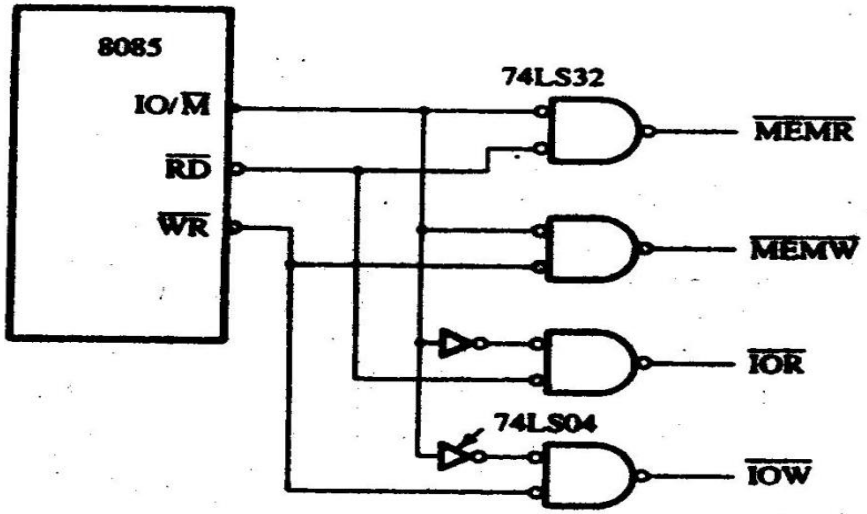
FIGURE 3.4
Schematic of Latching Low-Order Address Bus

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 $\overline{RD}$, $\overline{WR}$, and IO/$\overline{M}$.

FIGURE 3.5
Schematic to Generate Read/Write Control Signals for Memory and I/O



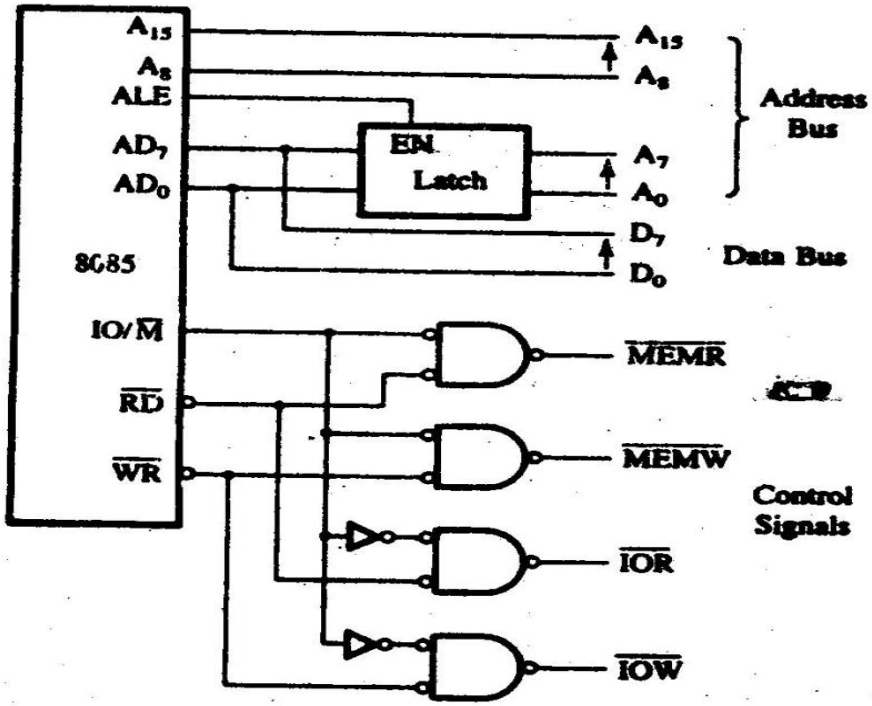
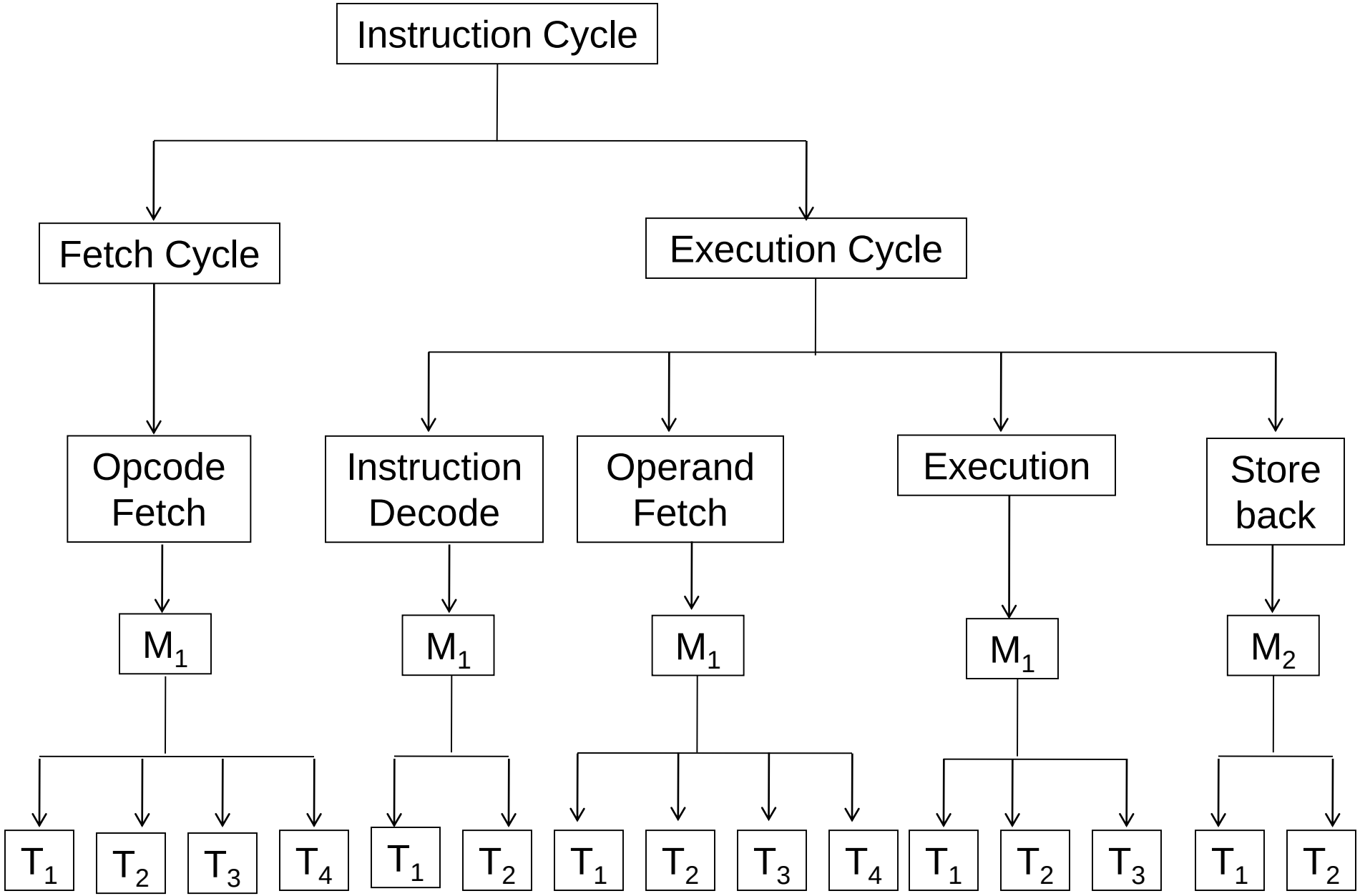


FIGURE 3.6
8085 Demultiplexed Address and Data Bus with Control Signals

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
Fetch Cycle
Execution Cycle
Opcode Fetch
Instruction Decode
Operand Fetch
Execution
Store back
M1
M1
M1
M1
M2
T1
T2
T3
T4
T1
T2
T1
T2
T3
T4
T1
T2
T3
T1
T2

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.

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

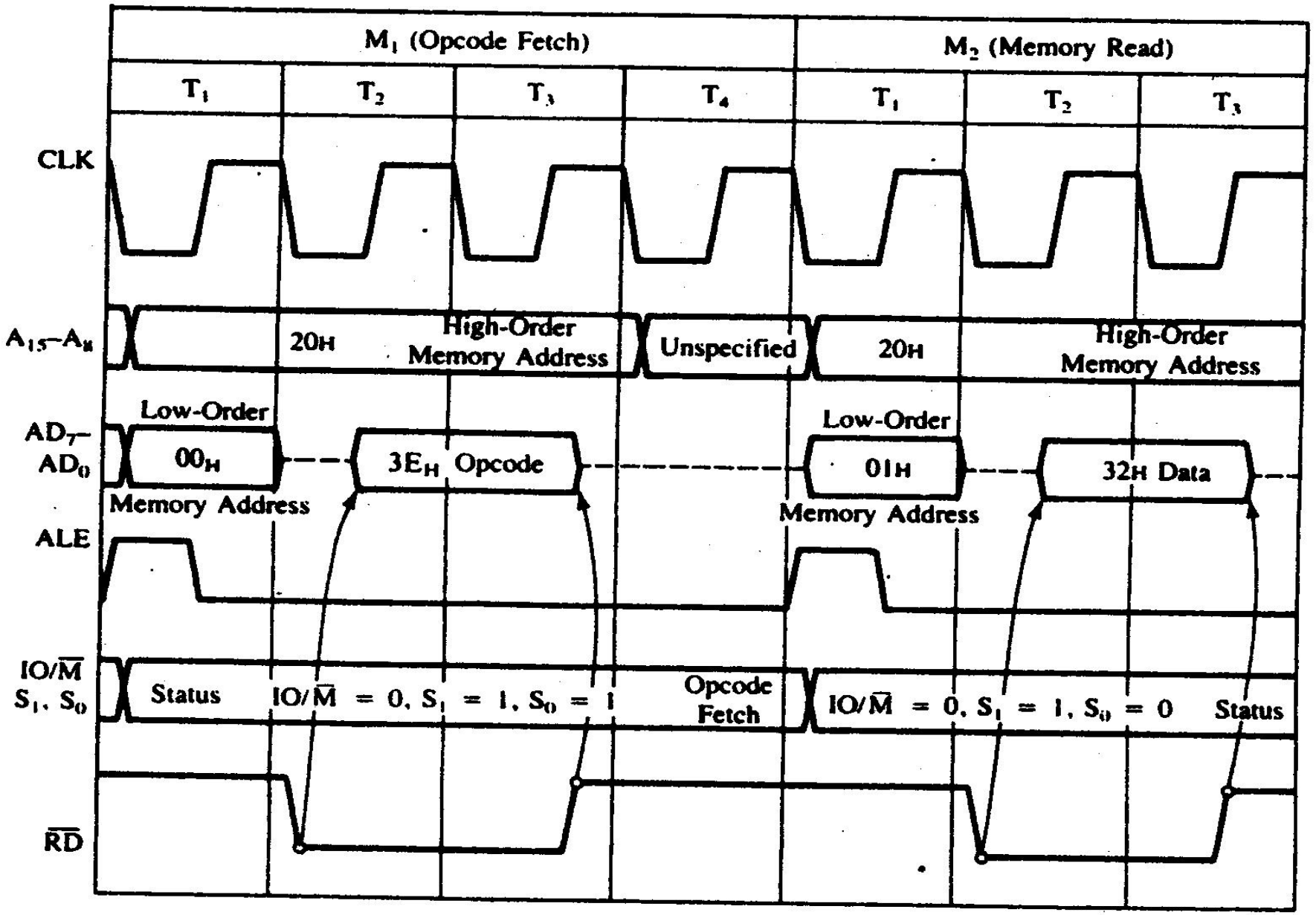
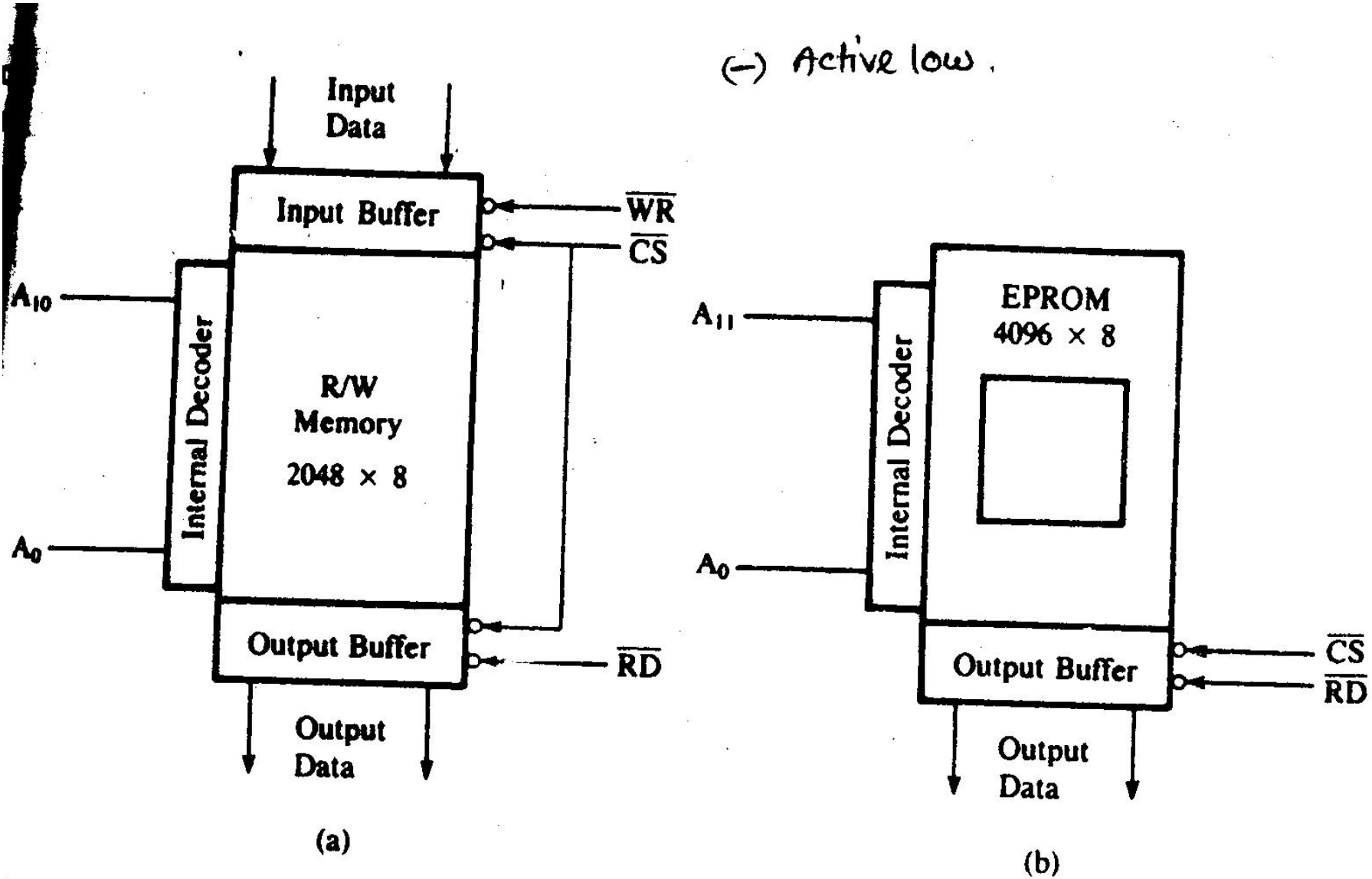


FIGURE 3.10
8085 Timing for Execution of the Instruction MVI A,32H

3.3 MEMORY INTERFACING

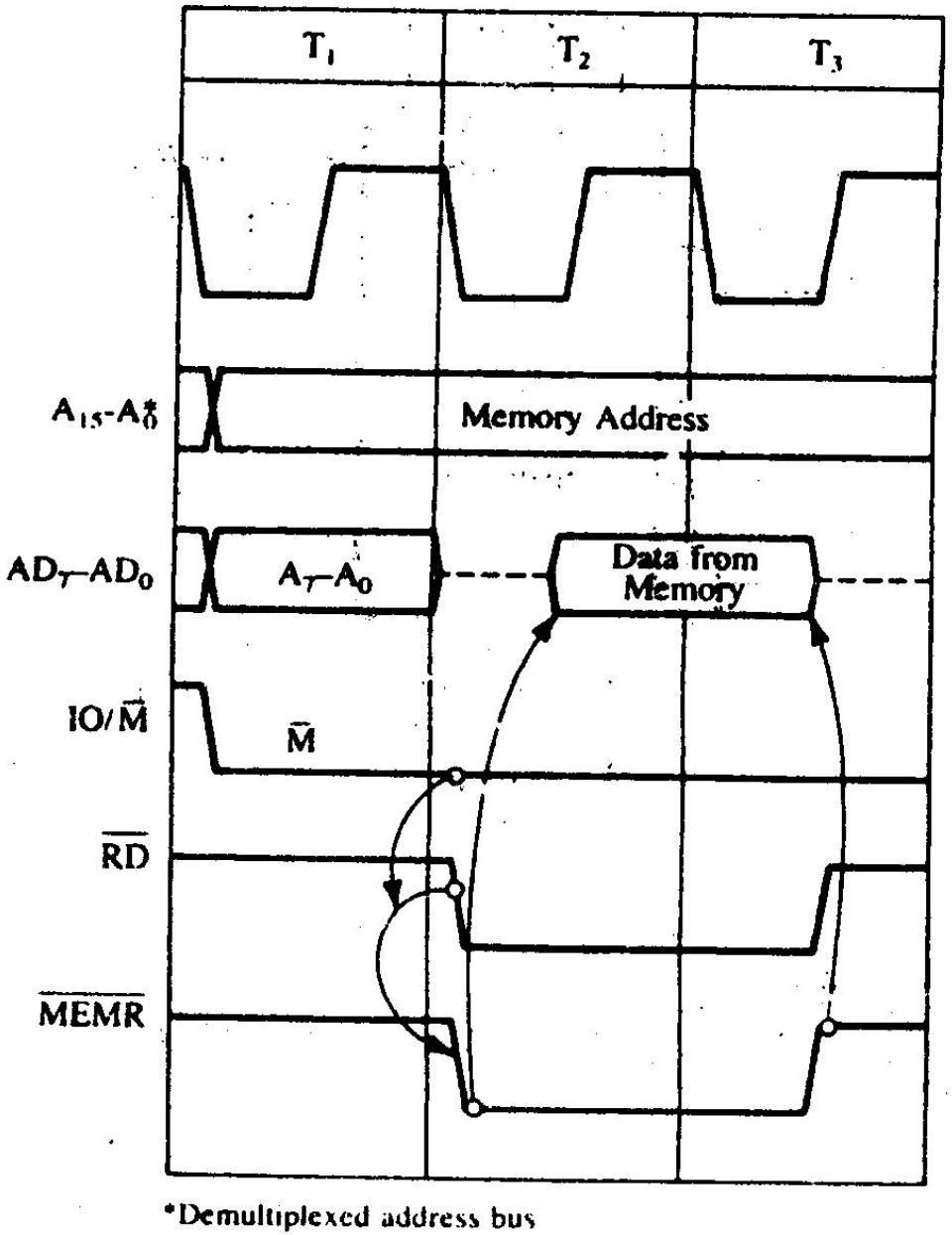
(-) Active low.



IGURE 3.11

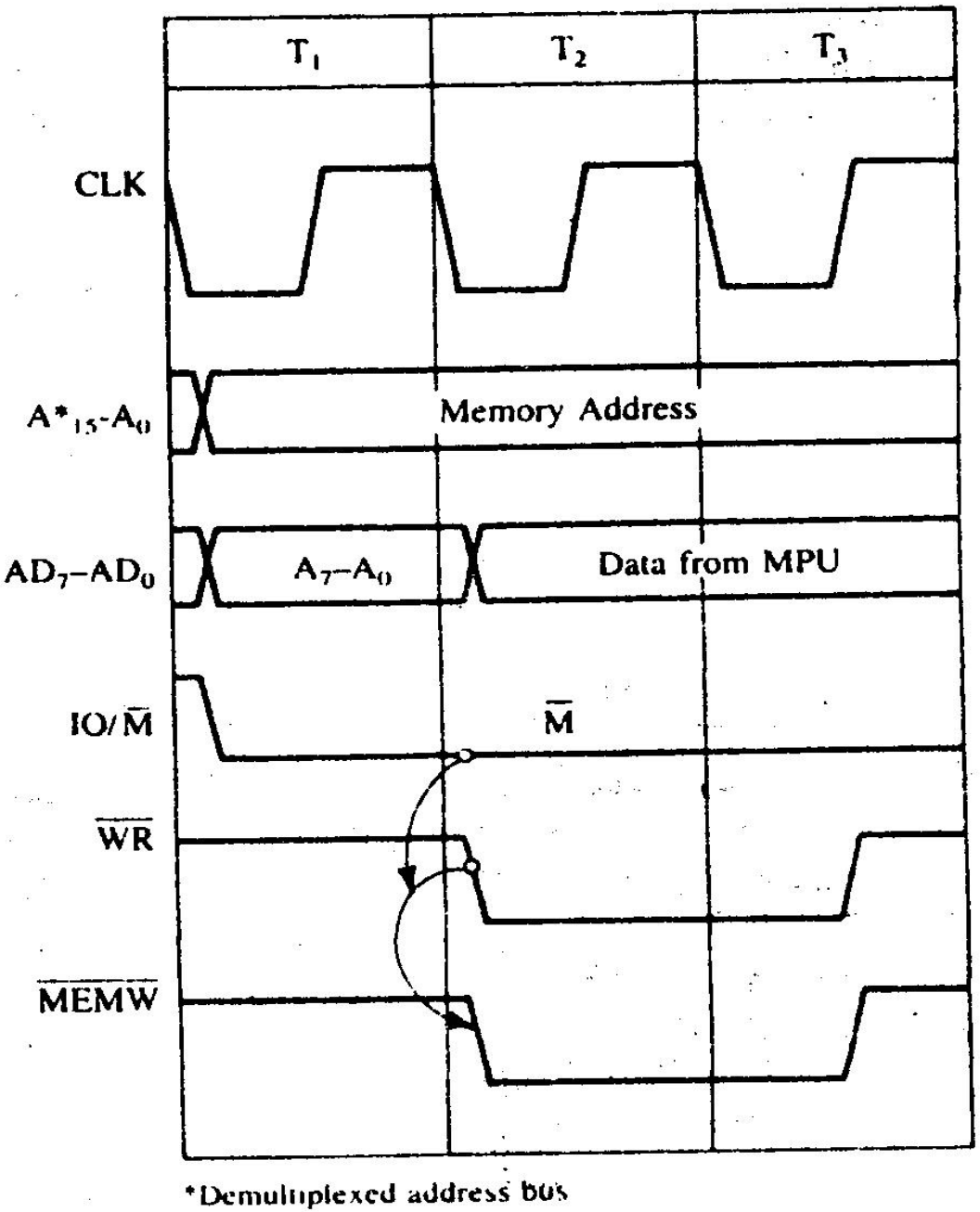
ypical Memory Chips: R/W Static Memory (a) and EPROM (b)

FIGURE 3.12
Timing of the Memory Read Cycle



*Demultiplexed address bus

FIGURE 3.13
Timing of the Memory Write Cycle



*Demultiplexed address bus

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

Mnemonic	Machine Code	Memory Address	Memory Contents
OUT 01H	D3	2050	11010011
	01	2051	00000001

4.1.2 I/O Execution

OUT INSTRUCTION

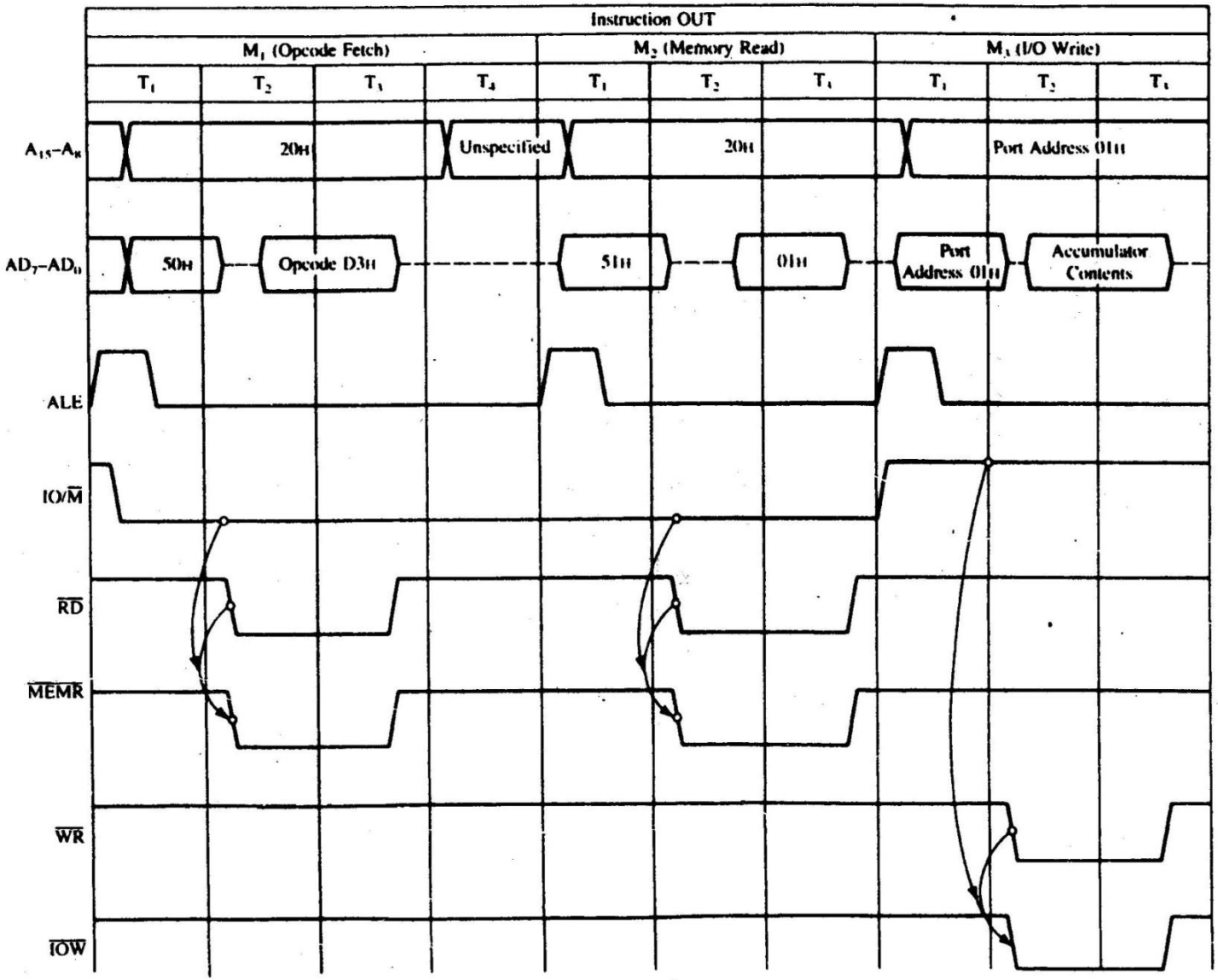


FIGURE 4.1
8085 Timing for Execution of OUT Instruction

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

Mnemonic	Machine Code	Memory Address	Memory Contents
IN 84H	DB	2065	11011011
	84	2066	10000100

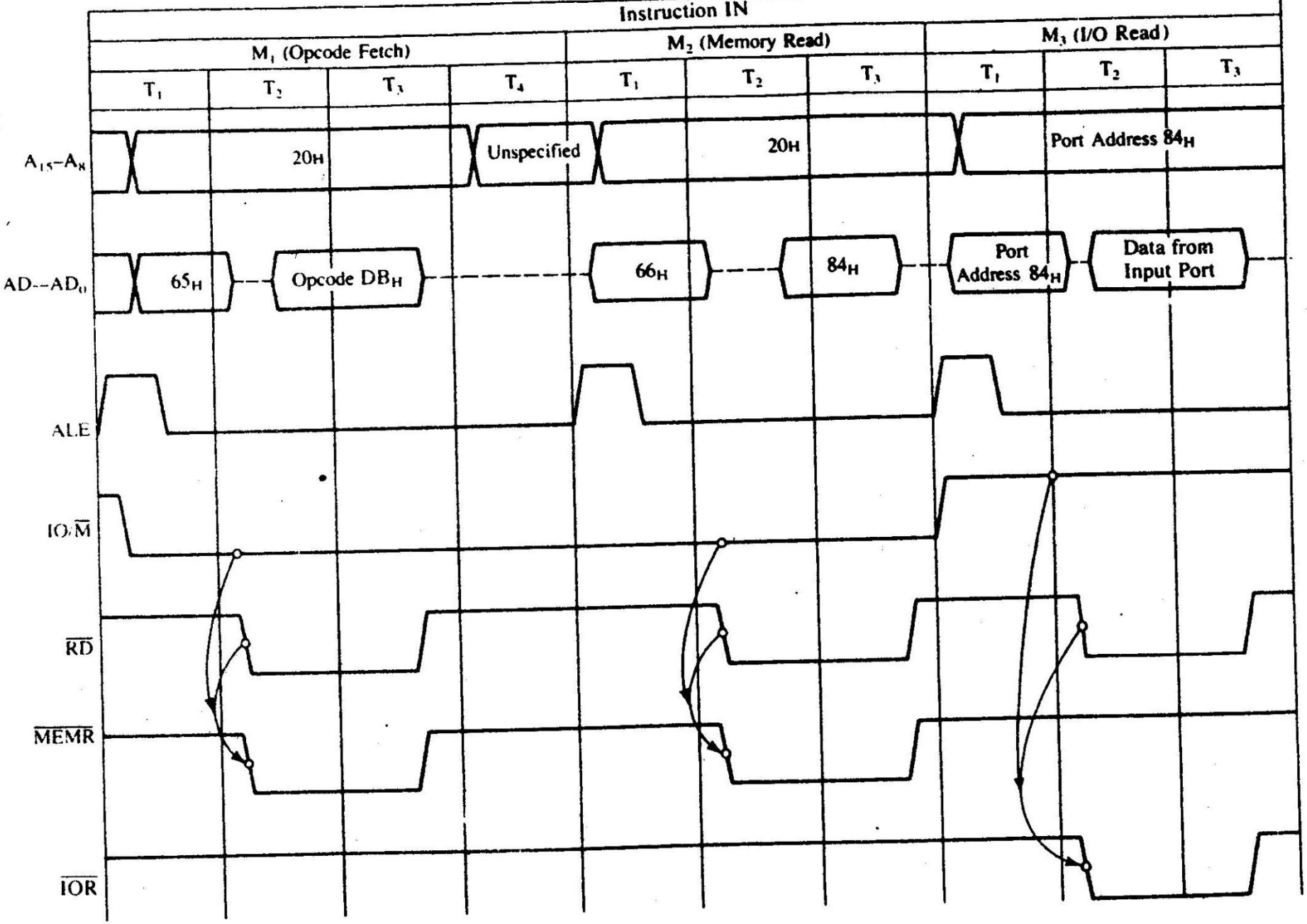


FIGURE 4.2
8085 Timing for Execution of IN Instruction

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.

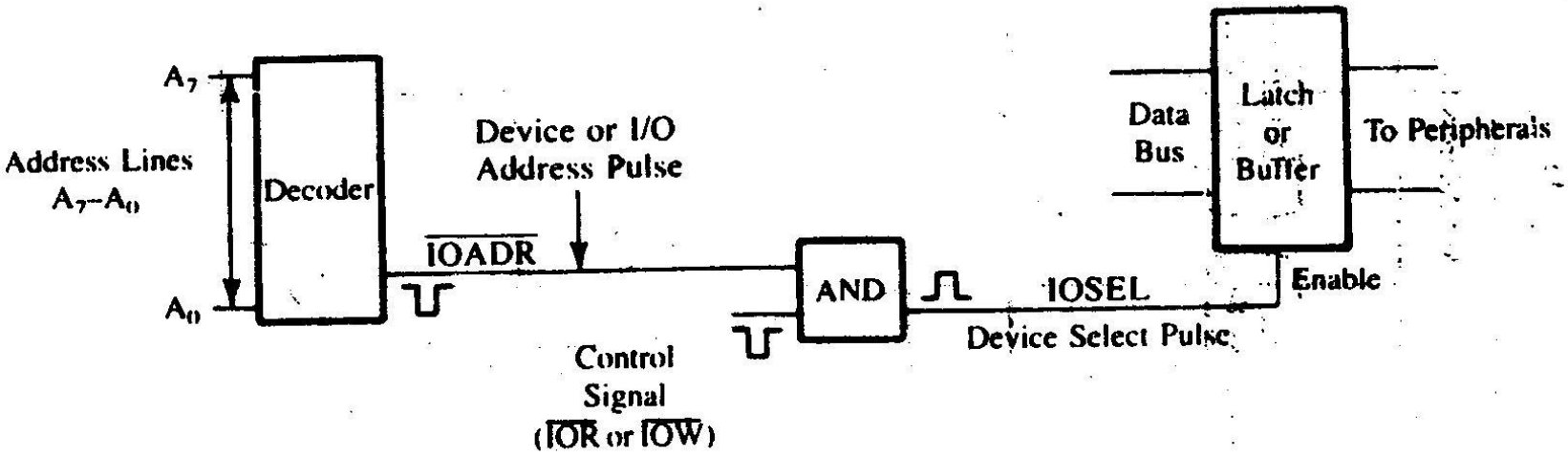


FIGURE 4.3
Block Diagram of I/O Interface

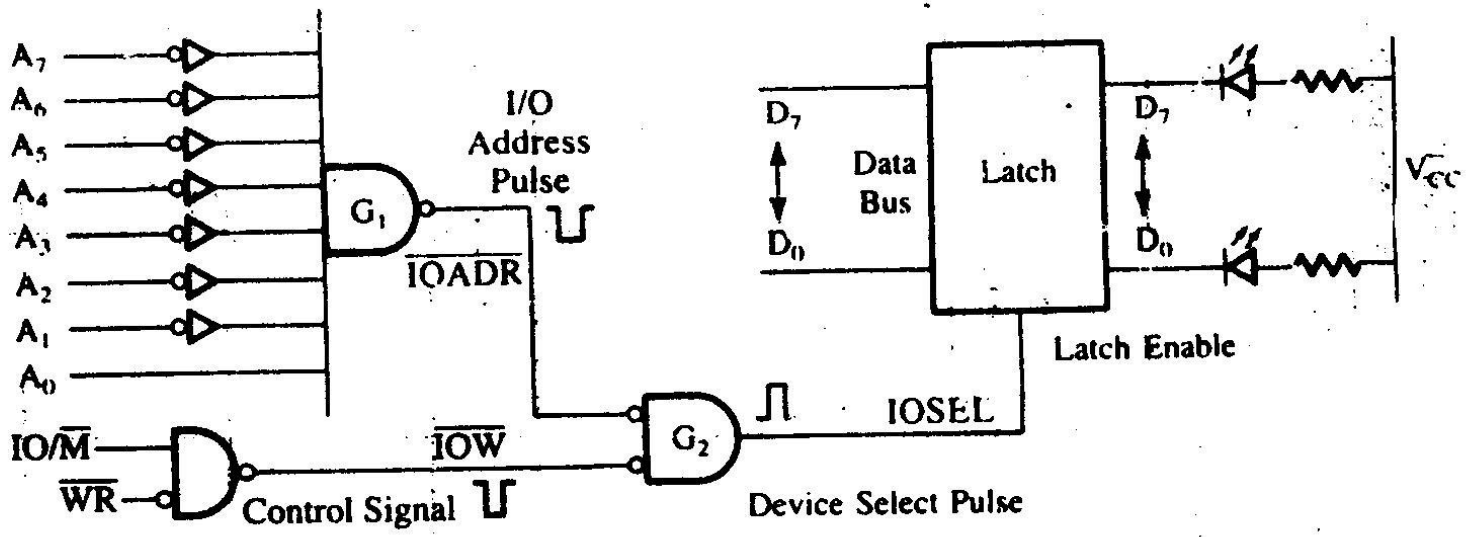


FIGURE 4.4
Decode Logic for LED Output Port

NOTE: To use this circuit with the 8085, the bus AD_7–A_0 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.

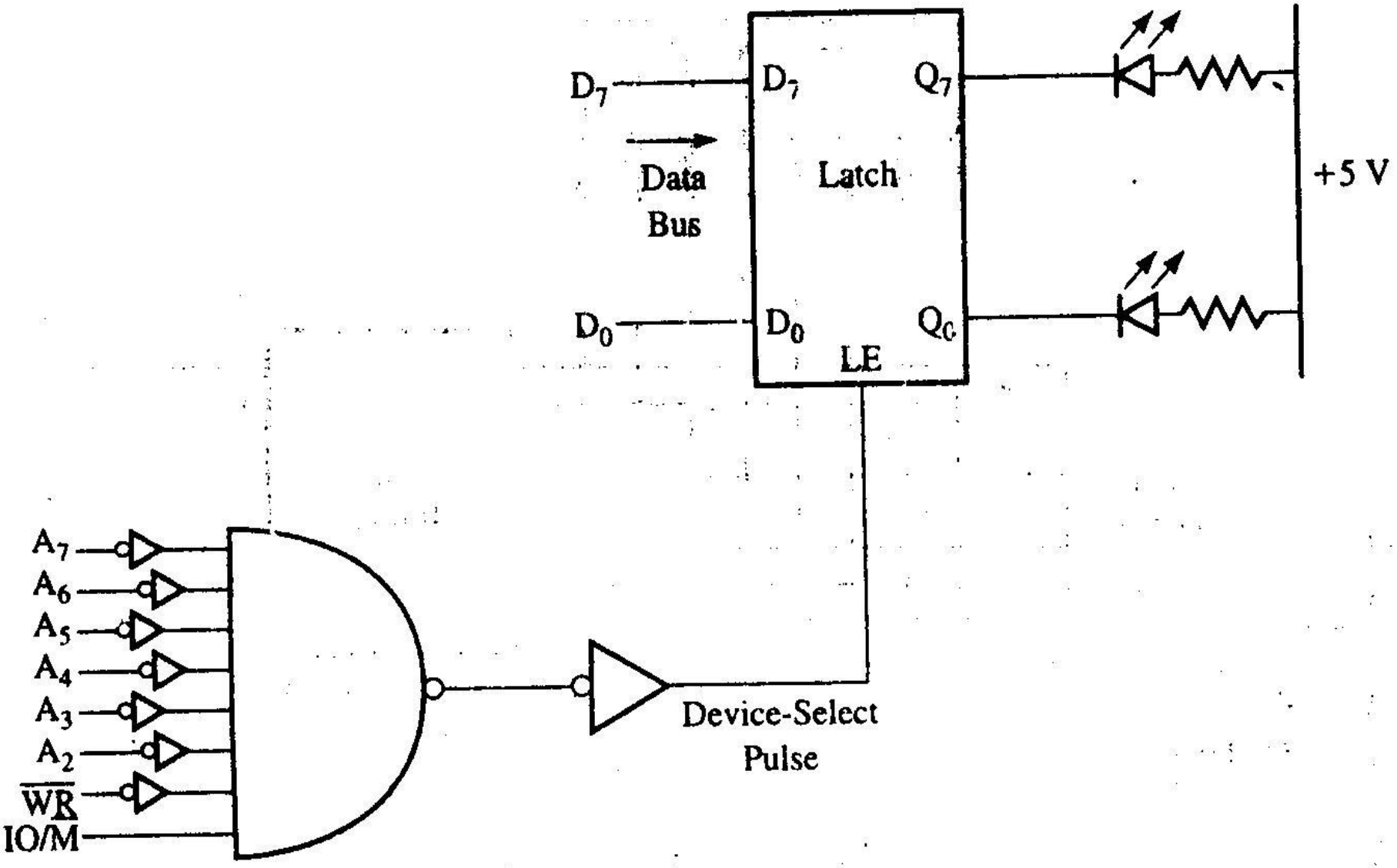


FIGURE 4.5
Partial Decoding: Output Latch with Multiple Addresses

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.

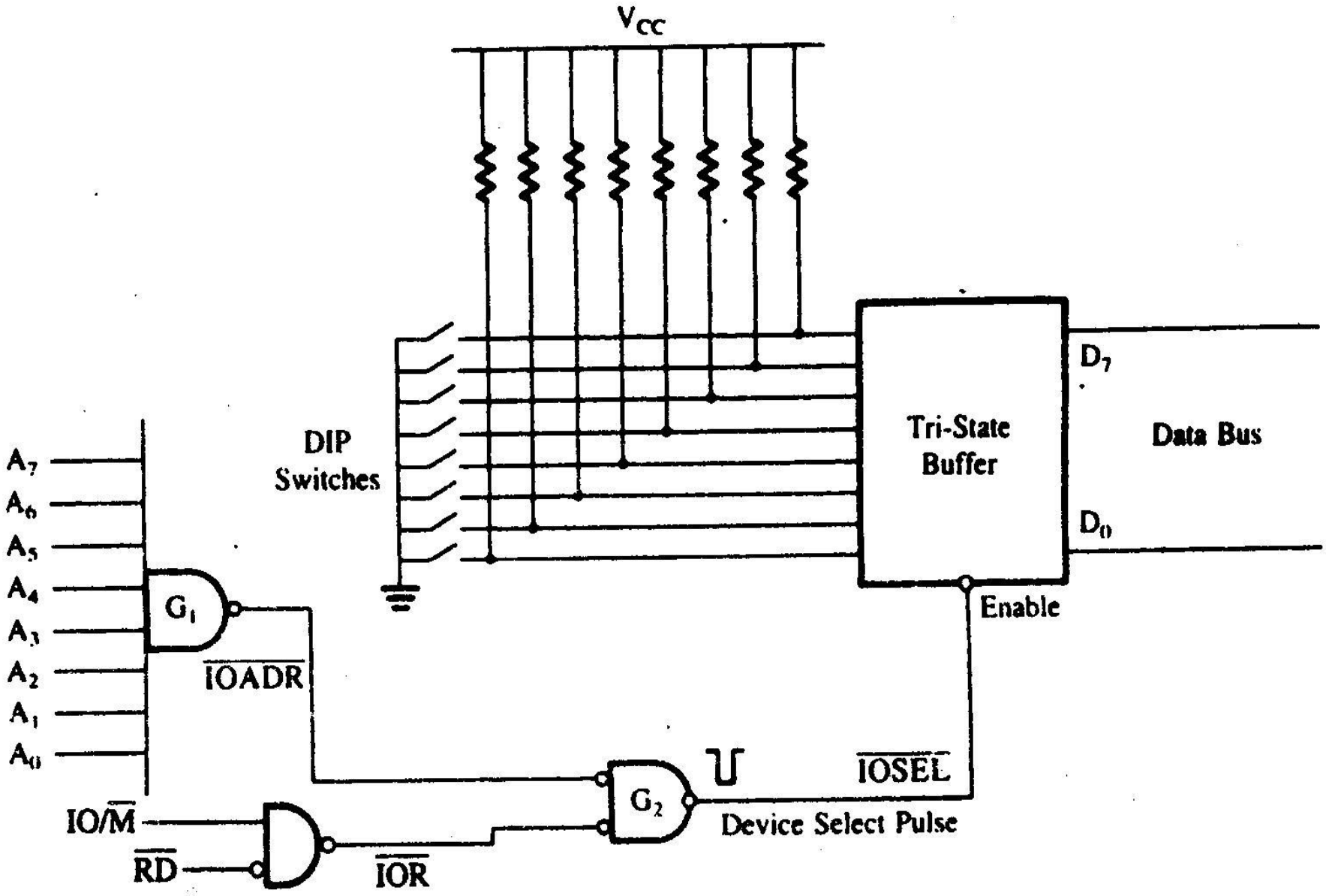


FIGURE 4.6
Decode Logic for a Dip-Switch Input Port

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)

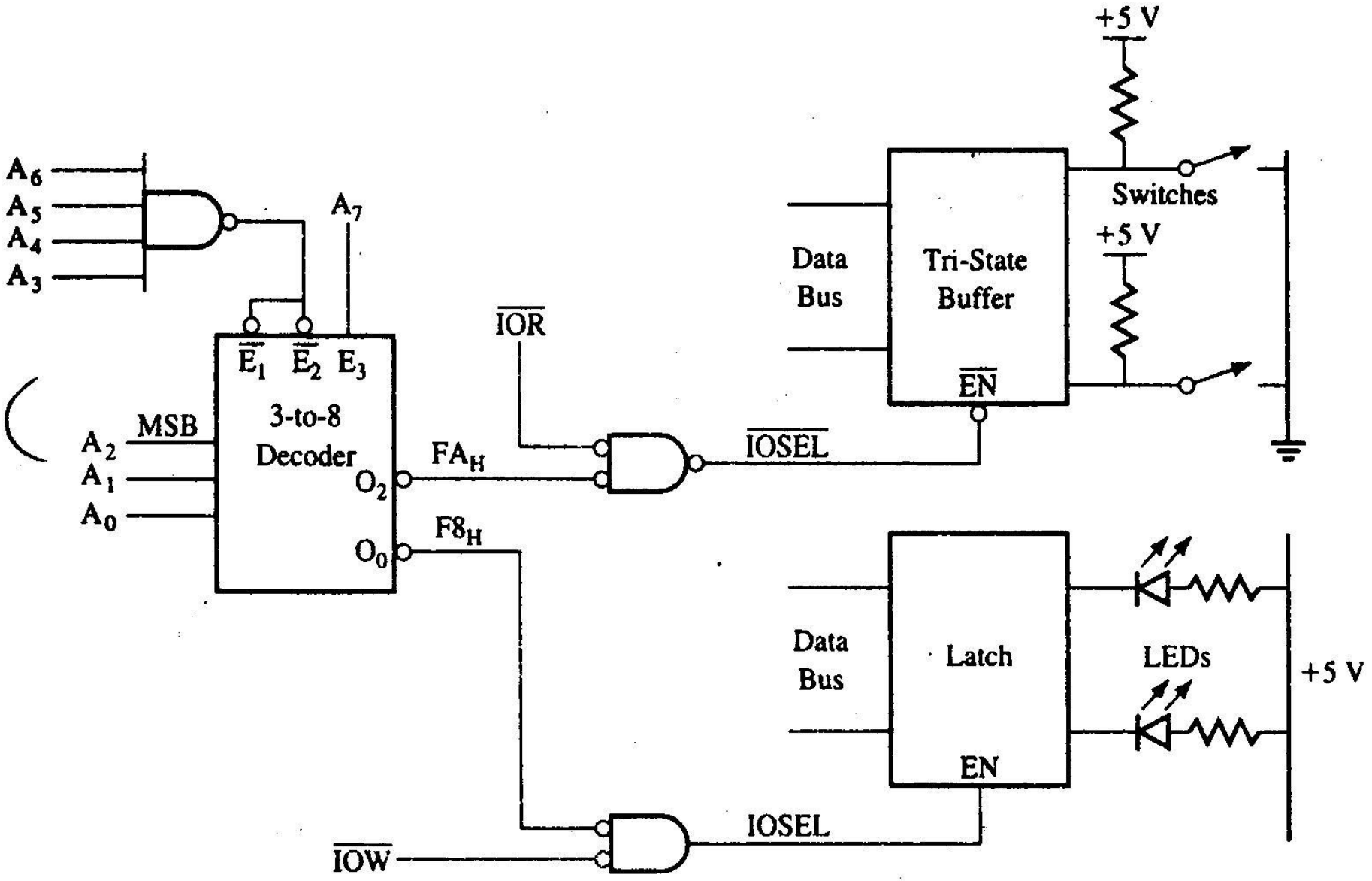


FIGURE 4.7
Address Decoding Using a 3-to-8 Decoder

4.2 INTERFACING OUTPUT DISPLAYS

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

PROBLEM STATEMENT

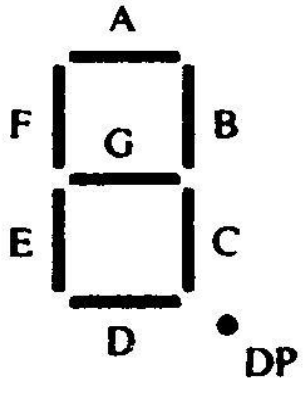
1. Design a seven-segment LED output port with the device address F5H, using a 74LS138 3-to-8 decoder, a 74LS20 4-input NAND gate, a 74LS02 NOR gate, and a common-anode seven-segment LED.
2. Given $\overline{WR}$ and IO/$\overline{M}$ signals from the 8085, generate the $\overline{IOW}$ control signal.
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

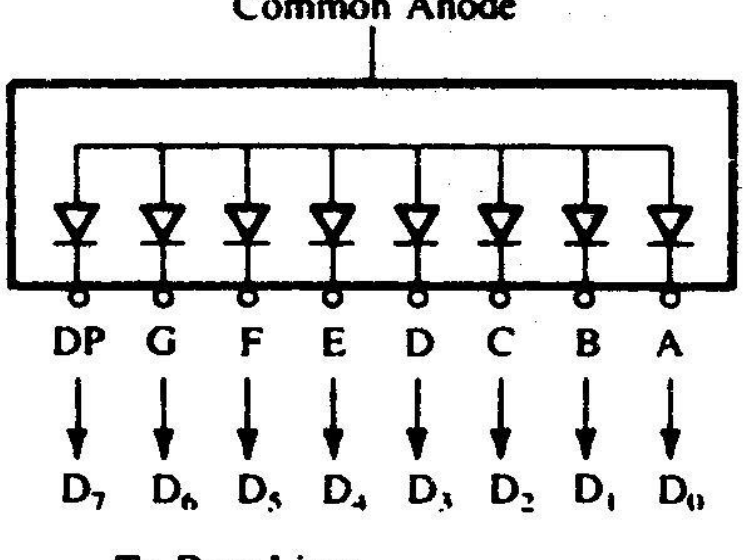
The circuit description is given below:

SEVEN-SEGMENT DISPLAY

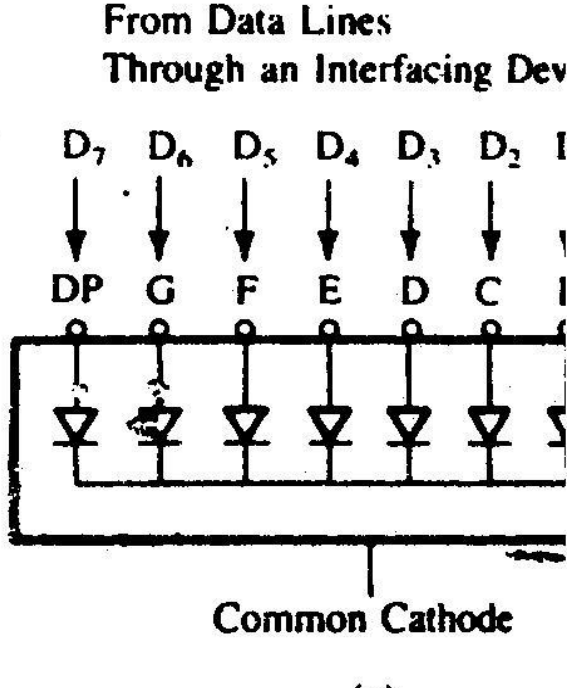
A seven-segment LED consists of seven light emitting diode and one segment for the decimal point. The LEDs are physically arranged as shown in Figure 4.9 (a). To display a number, the necessary segments are lit by sending an appropriate signals for current flow through diodes.



(a)



(b)



(c)

FIGURE 4.9

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

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	

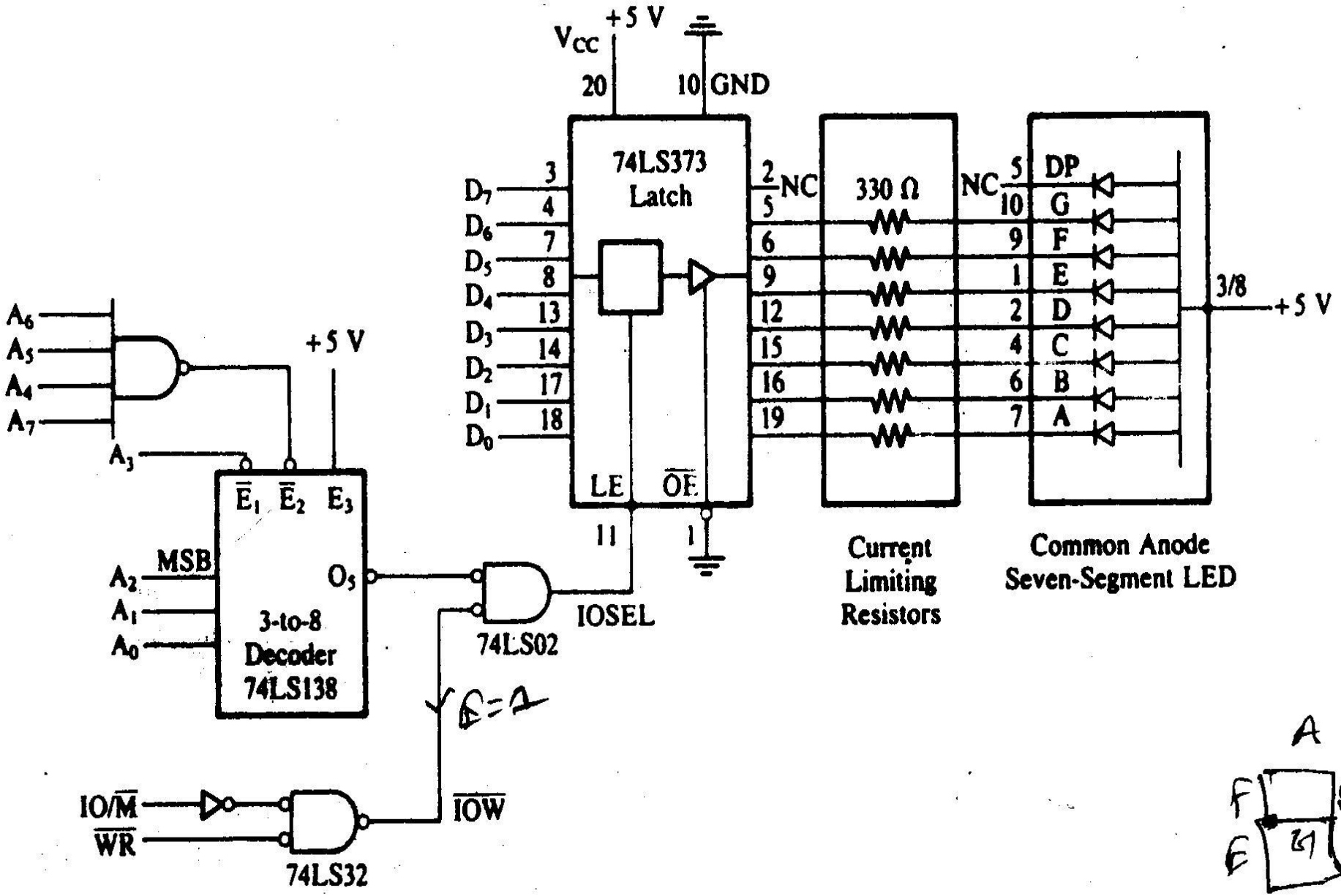


FIGURE 4.10
Interfacing Seven-Segment LED

7

To design an output port with the address F5H, the address lines A_7 – A_0 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 $\overline{OE}$ signal goes low, the input data show up on the output lines (connected to the data bus).

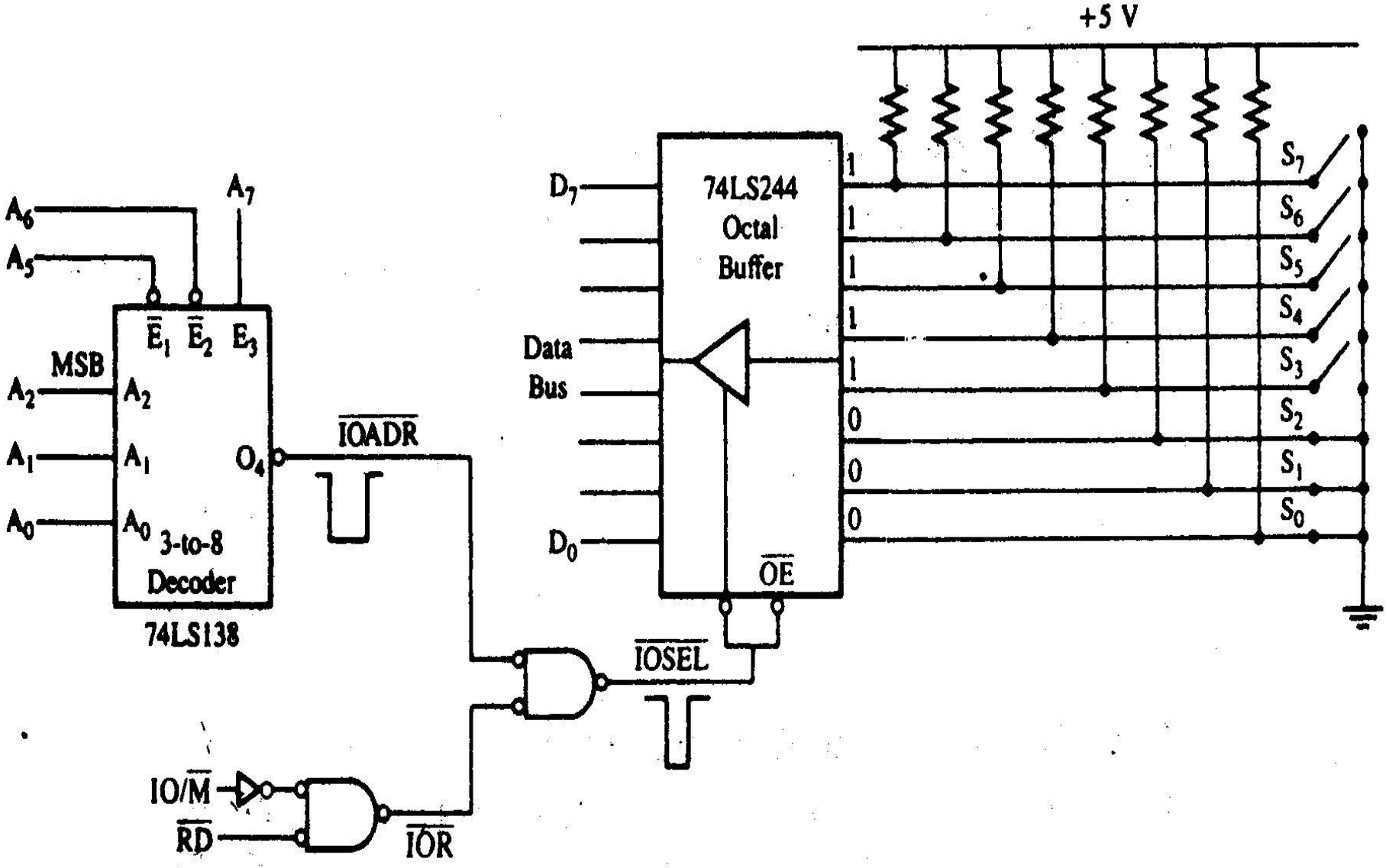


FIGURE 4.11
Interfacing DIP Switches

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
			0	1				= 8CH
			1	0				= 94H
			1	1				= 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 $\overline{RD}$ 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 $\overline{\text{MEMR}}$ and $\overline{\text{MEMW}}$ 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.

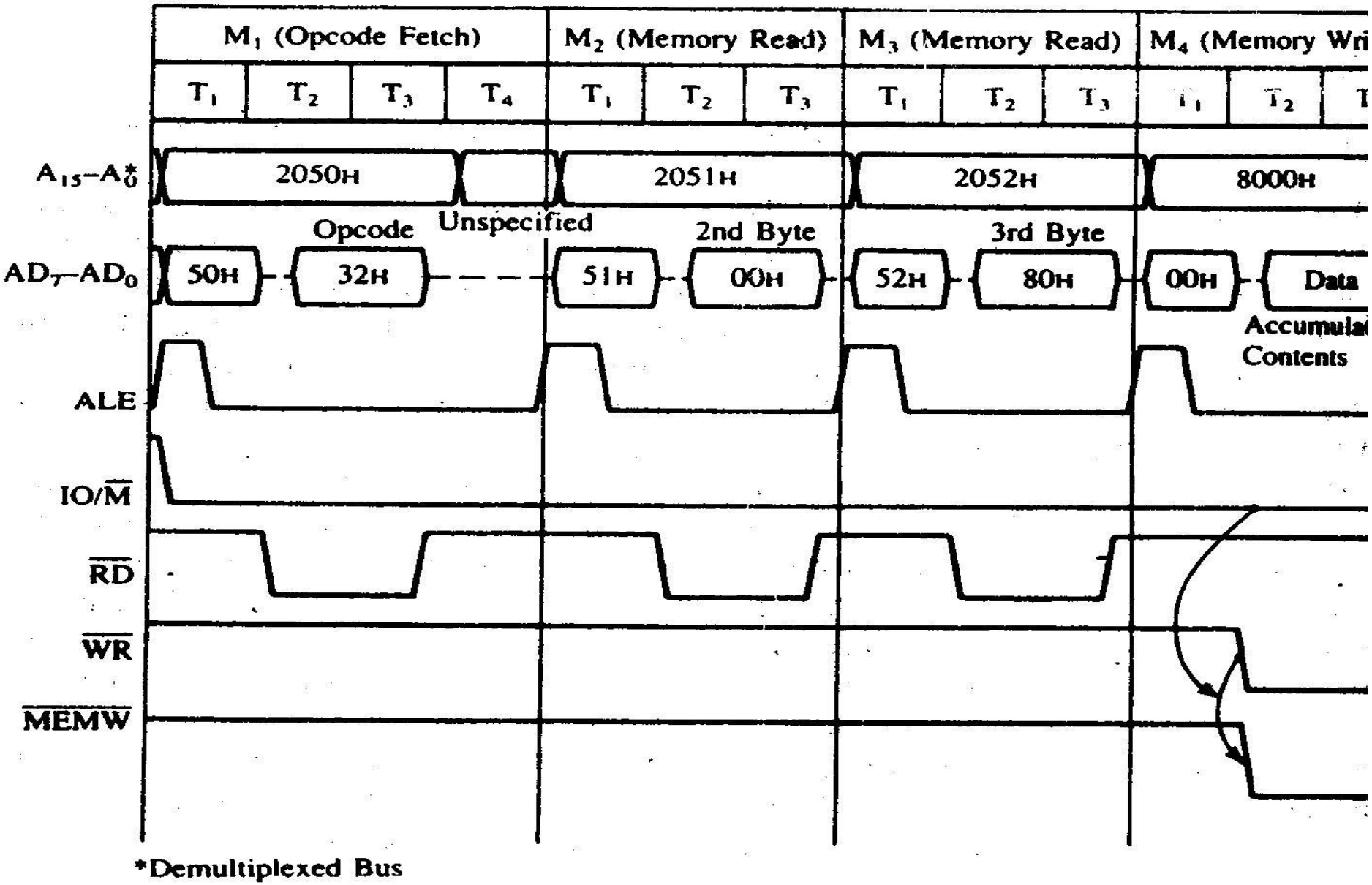


FIGURE 4.12
Timing for Execution of the Instruction: STA 8000H

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

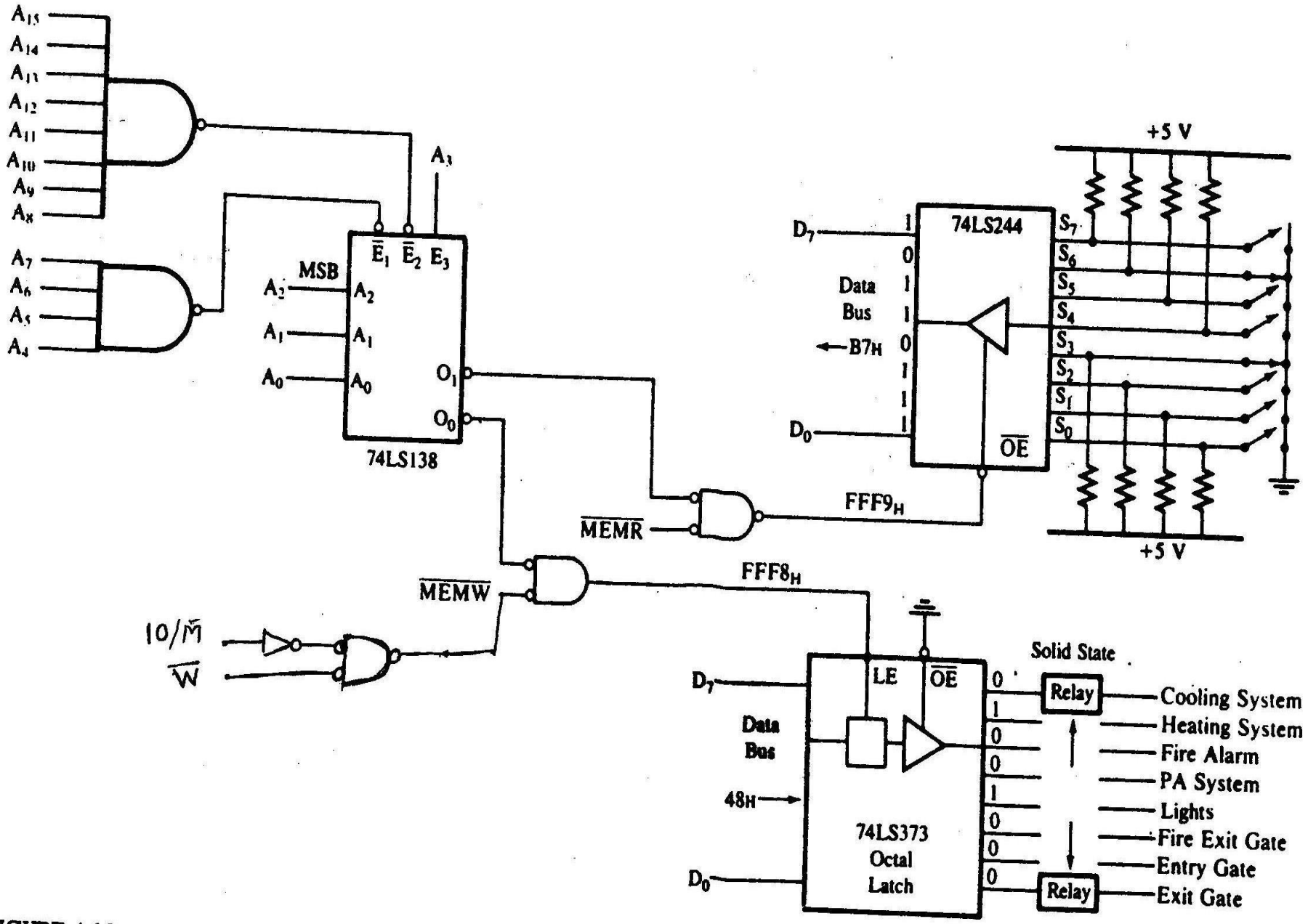


FIGURE 4.13
Memory-Mapped I/O Interfacing

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/OF 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
```

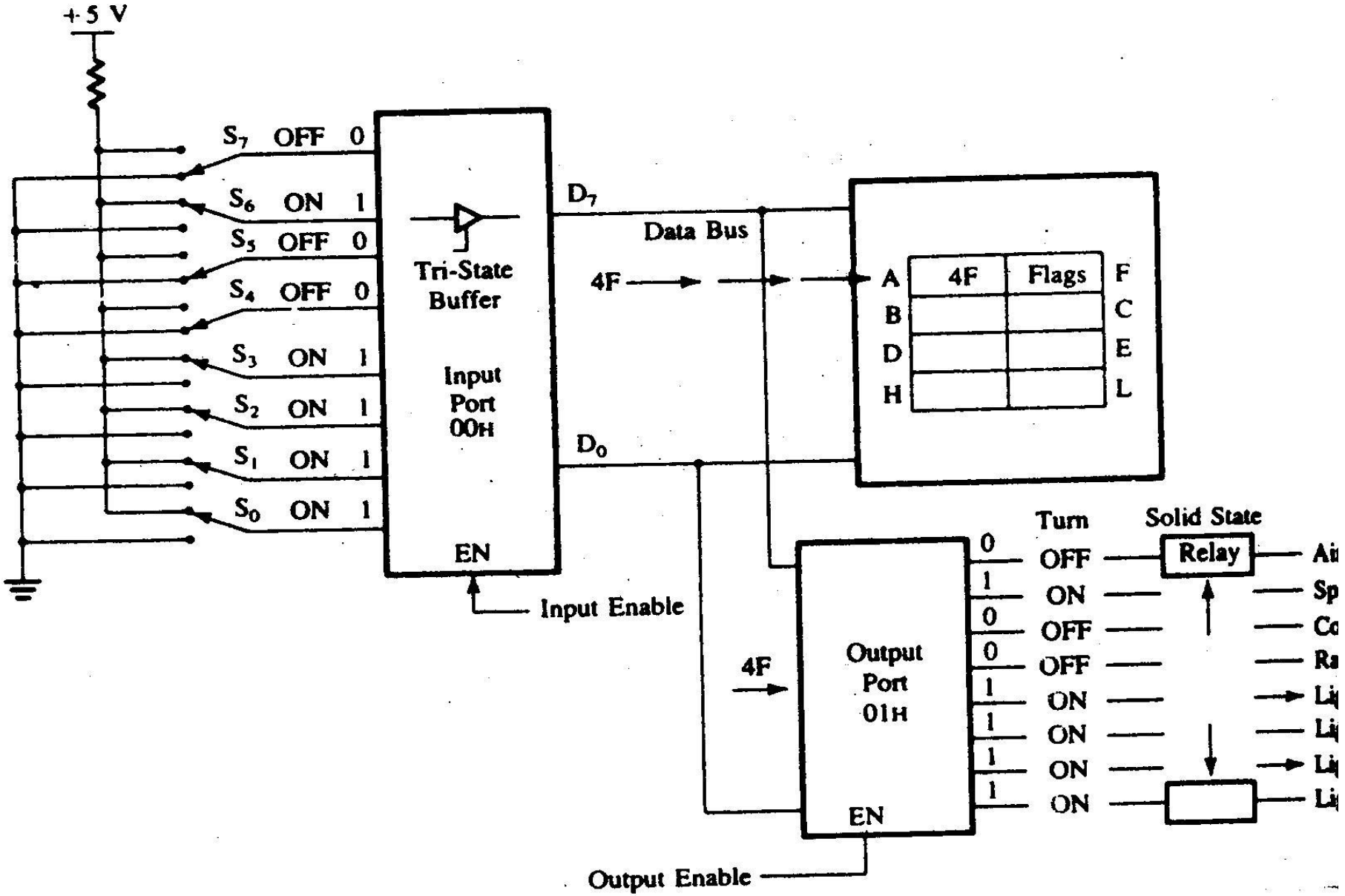


FIGURE 6.1
Reading Data at Input Port and Sending Data to Output Port

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:

IN 00H : (A) =	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
ORI 10H : =	0	0	0	1	0	0	0	0
(A) =	D_7	D_6	D_5	1	D_3	D_2	D_1	D_0

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

IN 00H : (A) =	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
ANI 7FH : =	0	1	1	1	1	1	1	1
(A) =	0	D_6	D_5	D_4	D_3	D_2	D_1	D_0

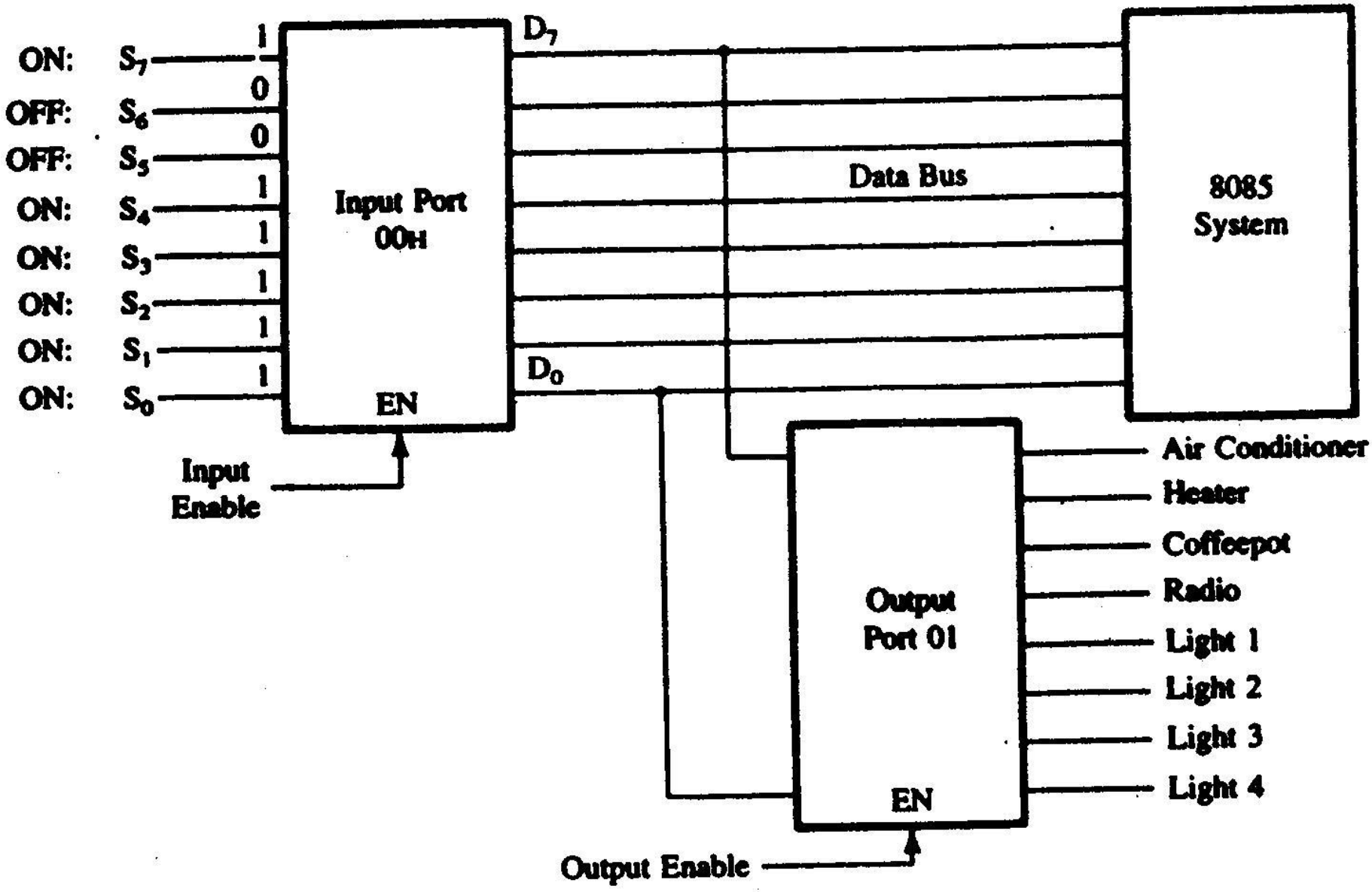


FIGURE 6.8
Input Port to Control Appliances

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.

PROGRM

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

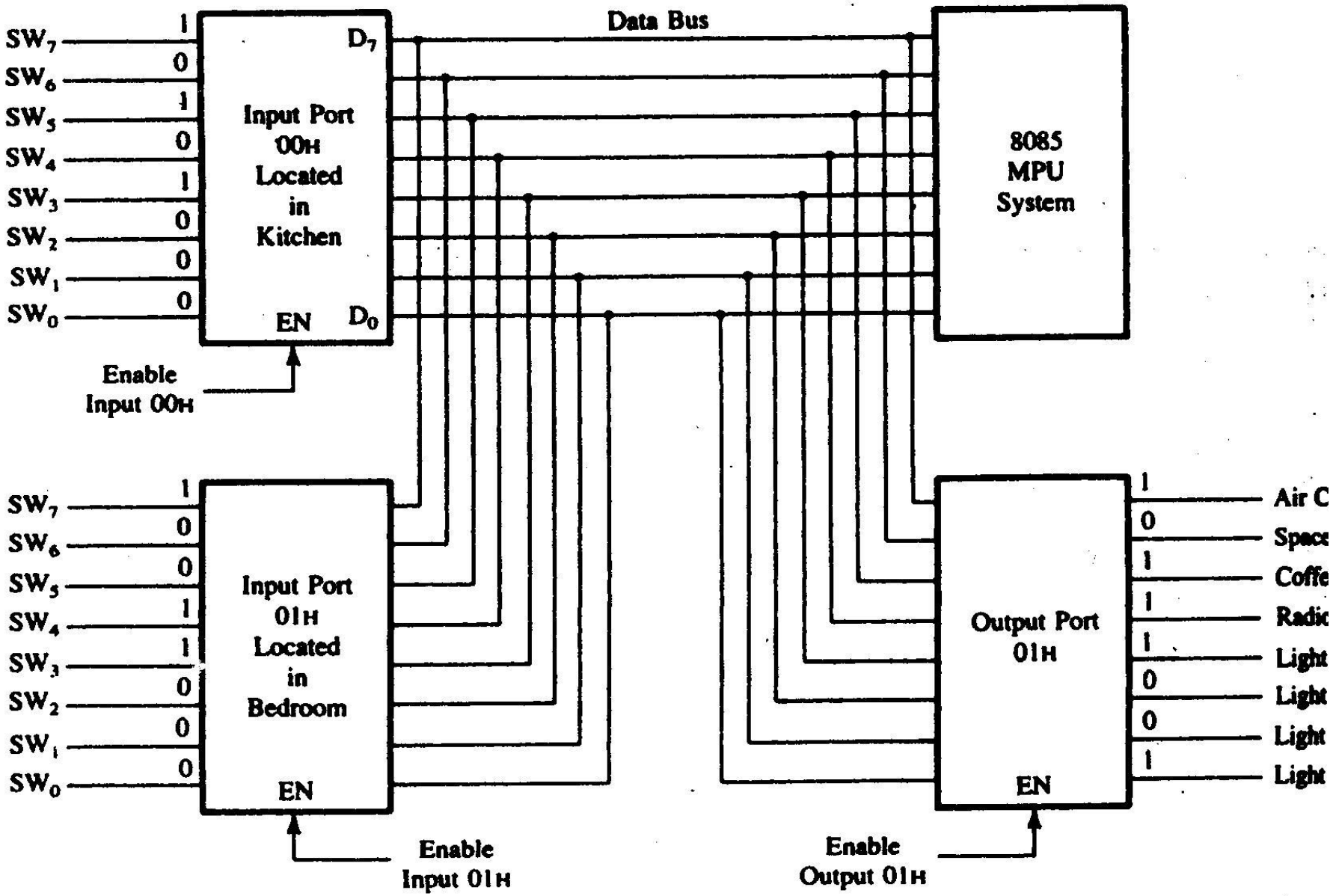


FIGURE 6.9
Two Input Ports to Control Output Devices

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.

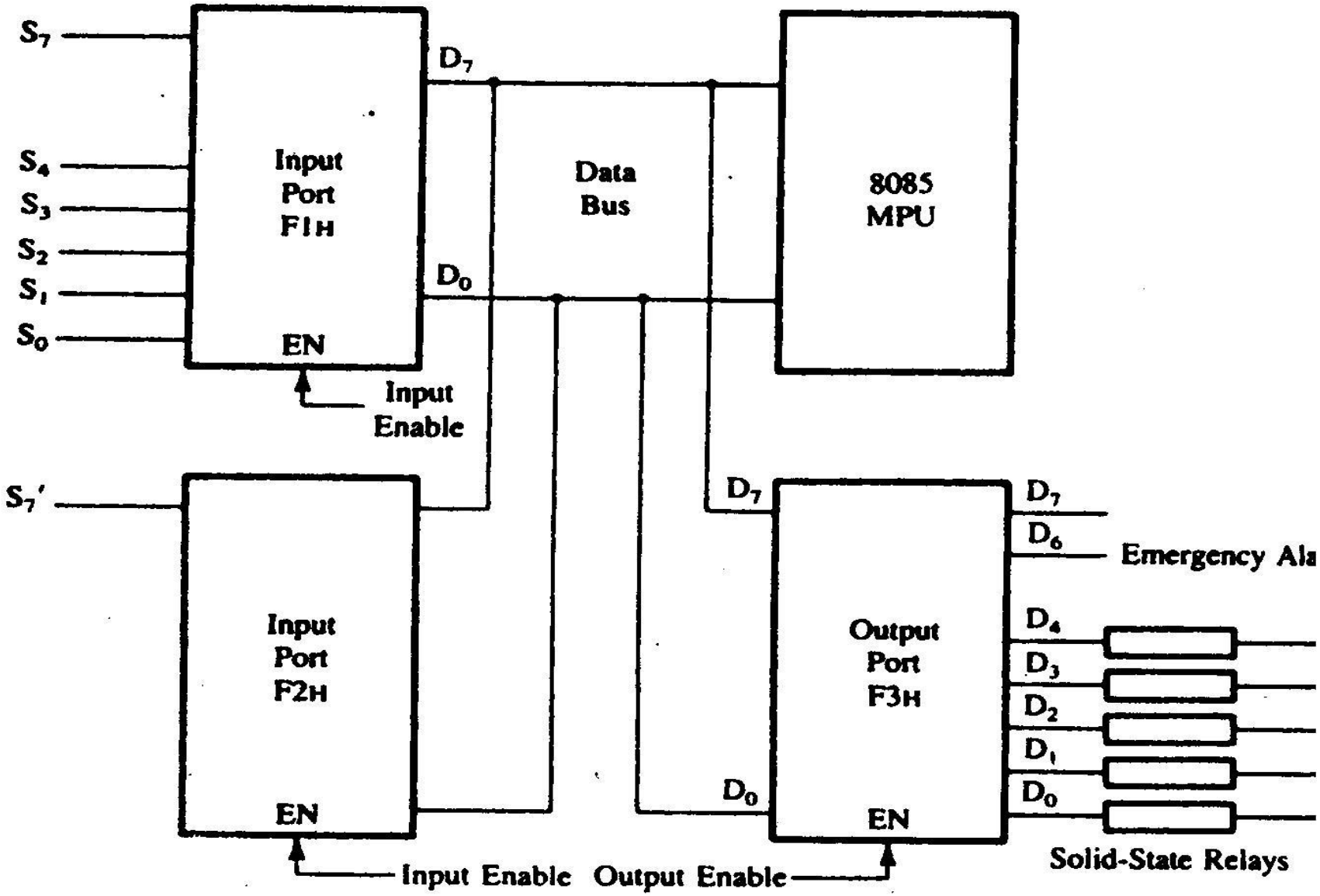


FIGURE 6.12
Input/Output Ports to Control Manufacturing Processes

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.

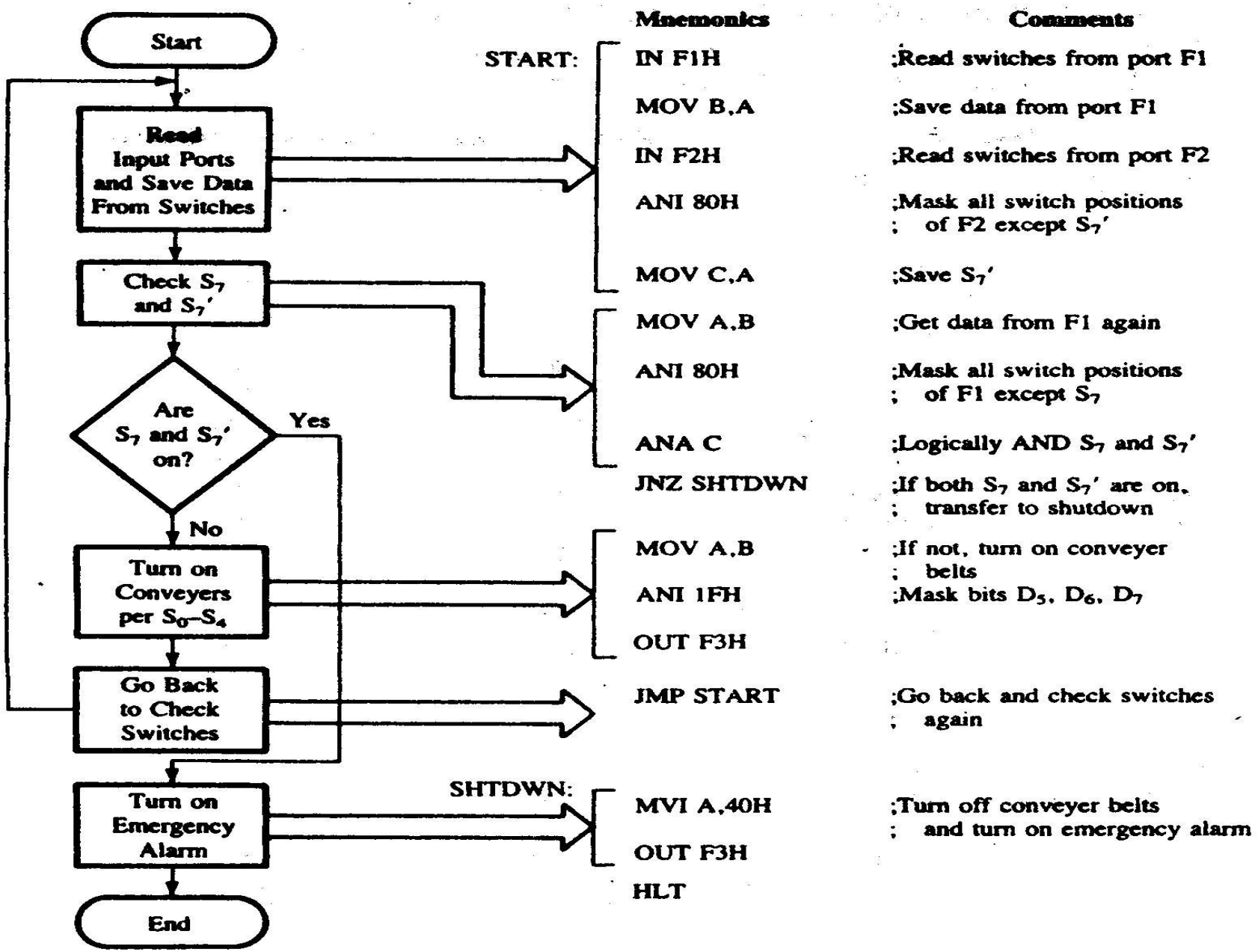


FIGURE 6.13
Flowchart and Program for Controlling Manufacturing Processes

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

Label	Opcode	Operand	T-states
	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 μs as follows:

Clock frequency of the system f = 2 MHz=2*10^6 Hz
Clock period T = 1/f = 1/2 x 10^6 = 0.5 μs=0.5*10^-6 s
Time to execute MVI = 7 T-states x 0.5
= 3.5 μs

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

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

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

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

Therefore, the total delay is

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

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

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.

Label	Opcode	Operand	T-states
	LXI	B,2384H	10
LOOP:	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 μs, the delay in the loop T_L is

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

Total Delay T_D = 109 ms + T_O

≈ 109 ms (the instruction LXI adds only 5 μs)

8.13 Time Delay Using a Loop within a Loop Technique

Label	Opcode	Operand	T-states
	MVI	B,38H	7
LOOP2:	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 μs. The counter value of LOOP2 is 56_{10} (38H). Delay for the LOOP2 is calculated as follows:

T_{L2} = 56 (T_{L1} + 21 T-states x 0.5 μs)

= 56 (1783.5 μs + 10.5 μs)

= 100.46 ms

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 T-states x T x Count

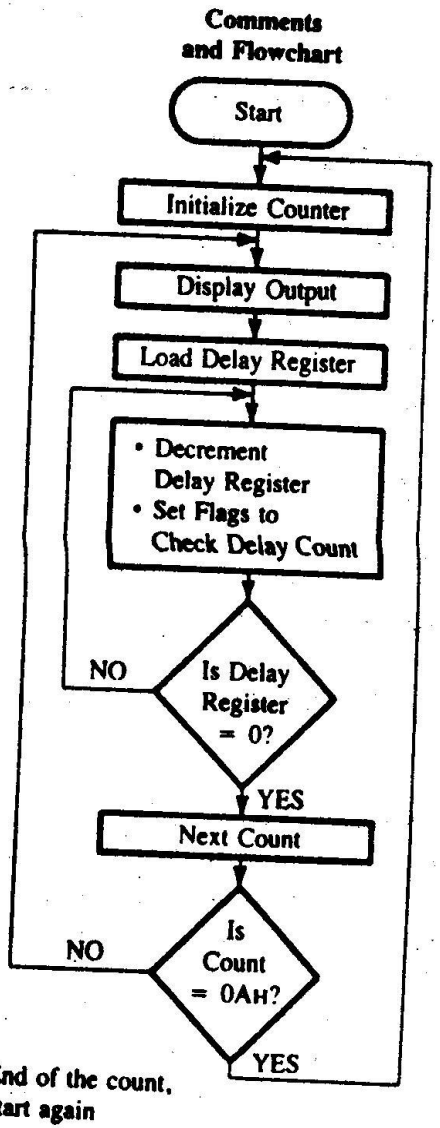
1 second = 24 x 1.0 x 10^{-6} x Count

Count = 1/(24 x 10^{-6})

= 41666_{10}

= A2C2H

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

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

Opcode	Operand
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

Opcode	Operand
CALL	16-bit memory address of a subroutine
RET	

9.21 Illustrative Program: Traffic Signal Controller

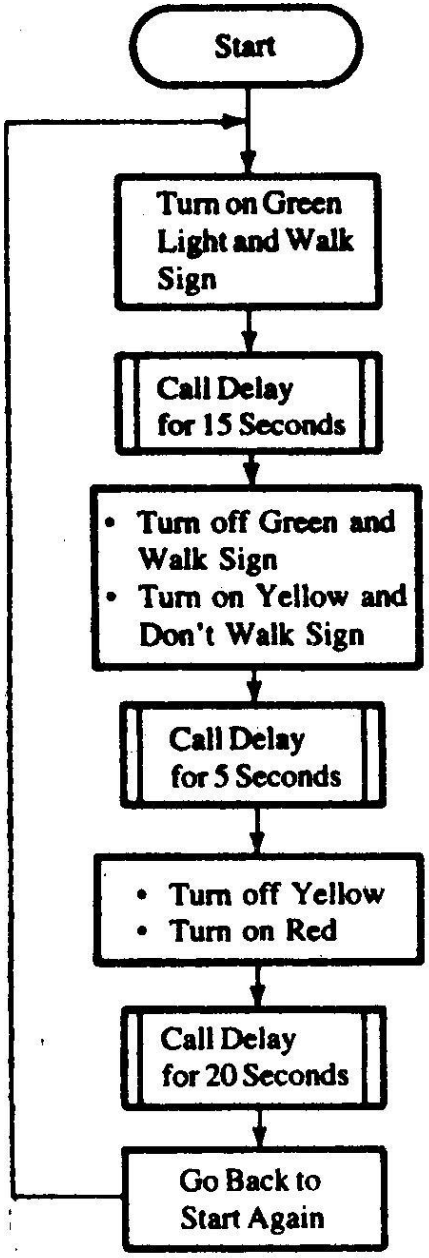
PROBLEM SATEMENT

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:

Lights	Data Bits	On Time
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

Time	DON'T WALK	WALK		Red		Yellow		Green	Hex
0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
↓ (15)									
15	0	1	0	0	0	0	0	1 =	41H
↓ (5)									
20	1	0	0	0	0	1	0	0 =	84H
↓ (20)									
40	1	0	0	1	0	0	0	0 =	90H


Start
Turn on Green Light and Walk Sign
Call Delay for 15 Seconds
• Turn off Green and Walk Sign
• Turn on Yellow and Don't Walk Sign
Call Delay for 5 Seconds
• Turn off Yellow
• Turn on Red
Call Delay for 20 Seconds
Go Back to Start Again

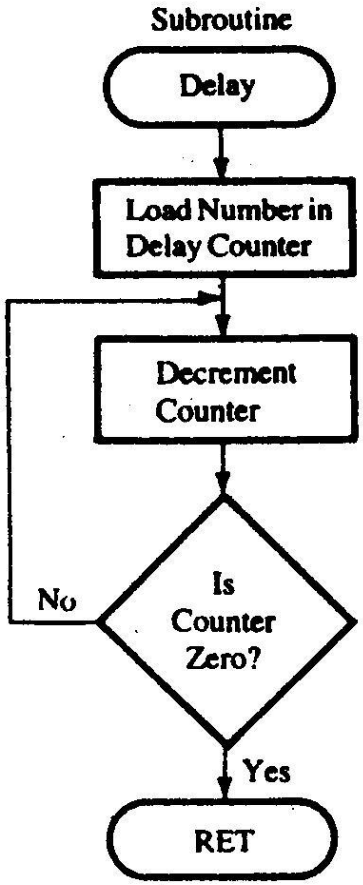

Subroutine
Delay
Load Number in Delay Counter
Decrement Counter
Is Counter Zero?
No
Yes
RET

FIGURE 9.13

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
```