UNIT-4

Stacks and Subroutines

Stacks:

A stack is a LIFO data structure which is employed in the RAM area /region of memory that is used to store data temporarily.

The 8085 has a 16-bit register known as the Stack Pointer. The function of the stack pointer is to hold the starting address of the stack. This address can be decided by the programmer.

The stack operates on the Last In, First Out (LIFO) principle. The location of the most recent data on the stack is known as the TOP of the stack. The stack pointer always points to the top of the stack. Contents can be stored in the stack using the PUSH instruction and can restore the contents by using the instruction POP

.

MNEMONIC	DESCRIPTION
LXI SP, 16-bit	Load the stack pointer register with a 16-bit address.
PUSH Rp	Copies the contents of the specified register pair on the stack
POP Rp	Copies the contents of the top two memory locations of the stack into the specified register pair.

PUSH and POP Operation in 8085

PUSH R_p

This is a 1-byte instruction. This instruction copies the contents of the specified register pair on the stack as described below:

- The stack pointer is decremented and the contents of the higher-order register are copied to the location shown by the stack pointer register.
- The stack pointer is again decremented and the contents of the low-order register are copied to that location.

POP R_p

This is a 1-byte instruction. This instruction copies the contents of the top two memory locations of the stack into the specified register pair.

- First, the contents of the memory location indicated by the stack pointer register are copied into the low-order register and then the stack pointer register is incremented by 1.
- The contents of the next memory location are copied into the high-order register and the stack pointer register is again incremented by 1.

Example

```
LXI SP,2099H
LXI H, 42F2H
PUSH H
Delay Counter
POP H
```

- The instruction
  LXI SP, *2099H* will initialize the stack pointer with the address of 2099H.
- LXI H, *42F2H* will initialize or load HL register pair with *42F2H* data so H = 42
  and L = F2
  .

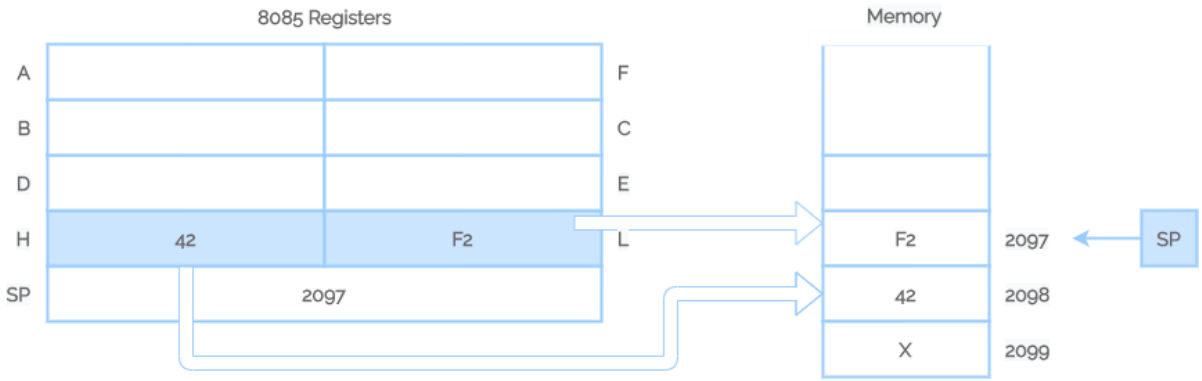
A			F
B			C
D			E
H	42	F2	L
SP	2099		

- After the execution of PUSH H instruction the stack pointer is decreased by one to *2098H* and the contents of the H register are copied to memory location *2098H*

.

- The stack pointer is again decreased by one to *2097H* and the contents of the L register are copied to memory location *2097H*

.

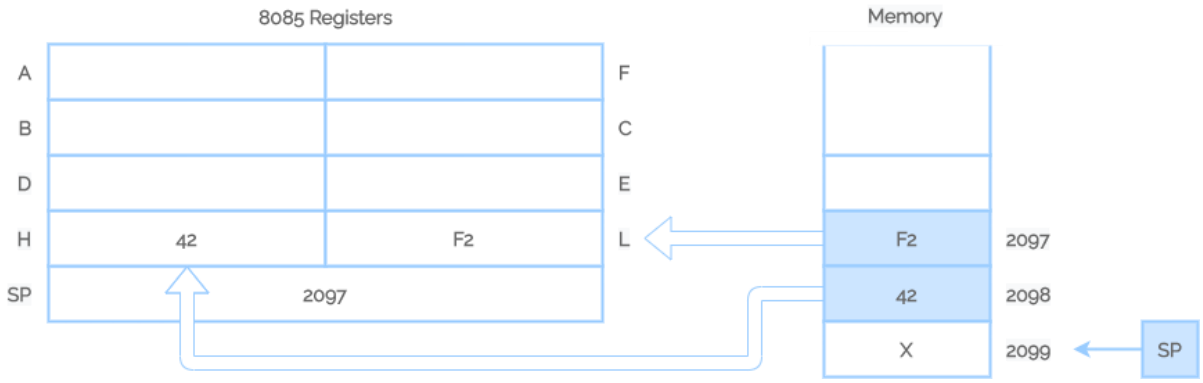


Contents of Stack and Registers After PUSH Operation

- After the execution of POP H instruction, the contents of the top of the stack location shown by the stack pointer are copied in the L register and the stack pointer is increased by one to *2098H*

.

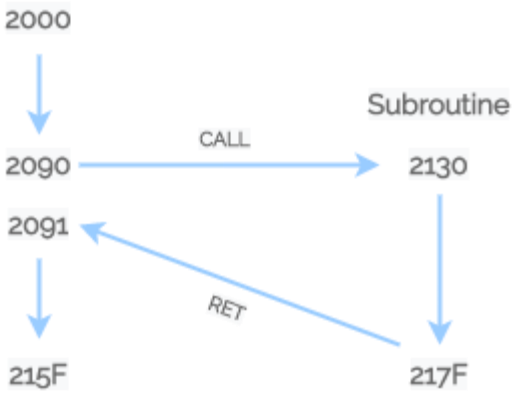
- The contents of the top of the stack are copied in the H register and the stack pointer is increased by one.
- The contents of the memory locations *2097H* and *2098H* are not destroyed until some other data bytes are stored in these locations.



Contents of Stack and Registers After POP Operation

Subroutines:

A subroutine is a set of instructions that will be used repeatedly in different locations of the program. Instead of repeating the same instructions several times, they can be combined into a subroutine that is called from various locations. A subroutine in Assembly language can exist anywhere in the code. However, It is a common practice to separate subroutines from the main program.



Subroutine Call and Program Transfer

The 8085 microprocessor has two instructions to implement subroutines. The CALL instruction is used to redirect program execution to a subroutine. The RET instruction is used to return the execution to the calling routine. They are described in the following table.

MNEMONIC	DESCRIPTION
CALL 16-bit address	3-byte instruction. Jumps unconditionally to the memory location specified.
RET	1-byte instruction. Unconditionally returns from the subroutine.

When the CALL instruction has been executed the contents of the program counter is saved to the top of the stack. This return address is retrieved from the stack when the RET instruction is executed.

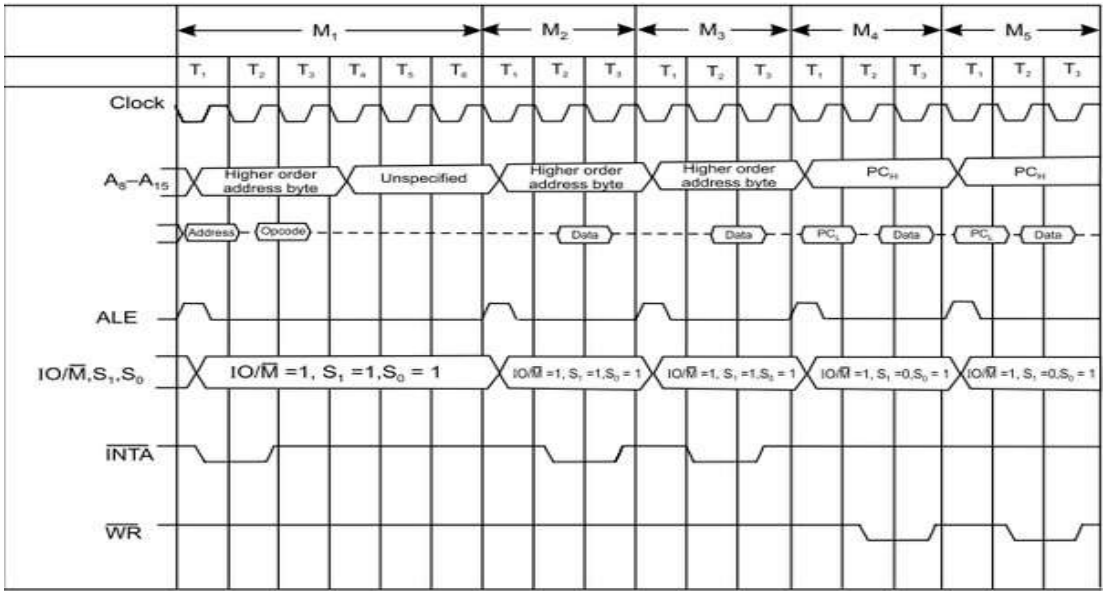
Let us consider the following sample code for a better explanation –

Address	Hex Codes	Mnemonic	Comment
2000	31	LXI SP, 5000H	SP ← 5000H.Initializing the SP
2001	00		Low order Byte of the address
2002	50		High order Byte of the address
2003	3E	MVI A, 00H	A ← 00H, Initializing the Accumulator
2004	00		00H as operand
2005	06	MVI B, 01H	B ← 01H
2006	01		01H as operand
2007	0E	MVI C, 02H	C ← 02H
2008	02		02H as operand

Address	Hex Codes	Mnemonic	Comment
2009	16	MVI D, 03H	D ← 03H
200A	03		03H as operand
200B	CD	CALL 2013H	Calling the subroutine at address 2013H. So now the control of the program will be transferred to the location2013H. And the return address 200EH i.e. address of the next instruction will be pushed on the top of the stack. As a result4FFFH (SP – 1) will contain 20H and 4FFEH (SP – 2) will contain 0EH respectively.
200C	13		Low order Byte of the address
200D	20		High order Byte of the address
200E	21	LXI H, 4050H	HL ← 4050H,Initializing the HL register pair. After execution of the RETinstruction, control will come back to this instruction. 4050Hwill have the value 06H, i.e. the final sum of 01H + 02H + 03H =06H
200F	50		Low order Byte of the address
2010	40		High order Byte of the address
2011	77	MOV M, A	M ← A, Content of the Accumulator will be transferred to the memory location 4050H as it is pointed by HL register pair

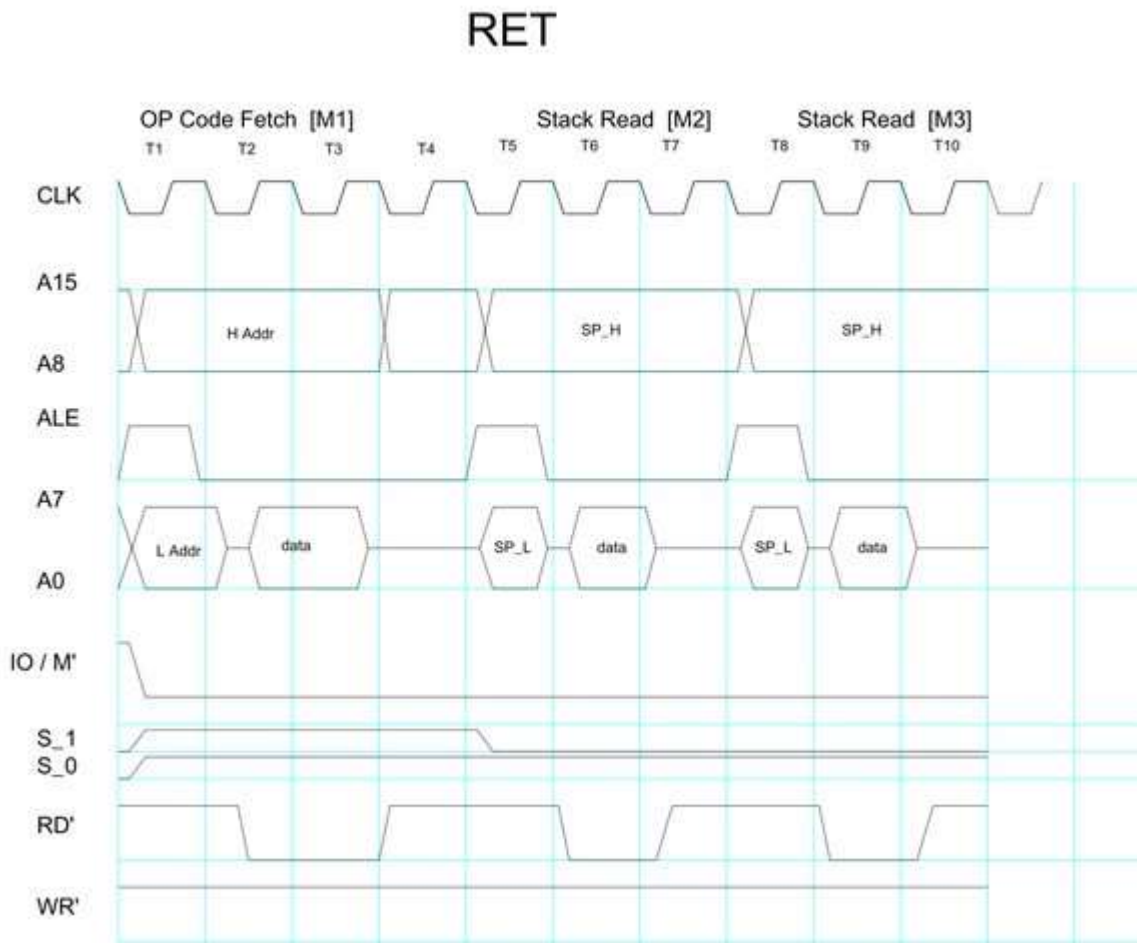
Address	Hex Codes	Mnemonic	Comment
2012	76	HLT	End of the program.
2013	80	ADD B	A ← A + B
2014	81	ADD C	A ← A + C
2015	82	ADD D	A ← A + D
2016	C9	RET	Return the control to the address 200EH. Return address 200EH will be popped out from the top of the stack. So from address 4FFEH, 0EH will be popped and from address 4FFFH 20Hwill be popped and SP will get the initial address 5000H back as its content accordingly.

The timing diagram against this instruction **CALL 2013H** execution is as follows –



Summary – So this instruction **CALL** requires 3-Bytes, 5-Machine Cycles (Opcode Fetch, Memory Read, Memory Read, Memory Write, Memory Write) and 18 T-States for execution as shown in the timing diagram.

The timing diagram against this instruction **RET** execution is as follows –



Summary – So this instruction **RET** requires 1-Byte, 3-Machine Cycles (Opcode Fetch, Memory Read, Memory Read) and 10 T-States for execution as shown in the timing diagram.

UNIT-4

Basic Interfacing Concepts

Input Output Interfacing 8085 Microprocessor:

Any application of a microprocessor-based system requires the transfer of data between external circuitry to the microprocessor and microprocessor to the external circuitry. User can give information to the microprocessor using keyboard and user can see the result or output information from the microprocessor with the help of display device. The transfer of data between keyboard and microprocessor, and microprocessor and display device is called Input Output Interfacing 8085 Microprocessor or I/O data transfer. This data transfer is done with the help of I/O ports.

Input Port:

It is used to read data from the input device such as keyboard. The simplest form of input port is a buffer. The input device is connected to the microprocessor through buffer as shown in the Fig. 4.28. This buffer is a tri-state buffer and its output is available only when enable signal is active.

When microprocessor wants to read data from the input device (keyboard), the control signals from the microprocessor activates the buffer by asserting enable input of the buffer. Once the buffer is enabled, data from the input device is available on the data bus. Microprocessor reads this data by initiating read command.

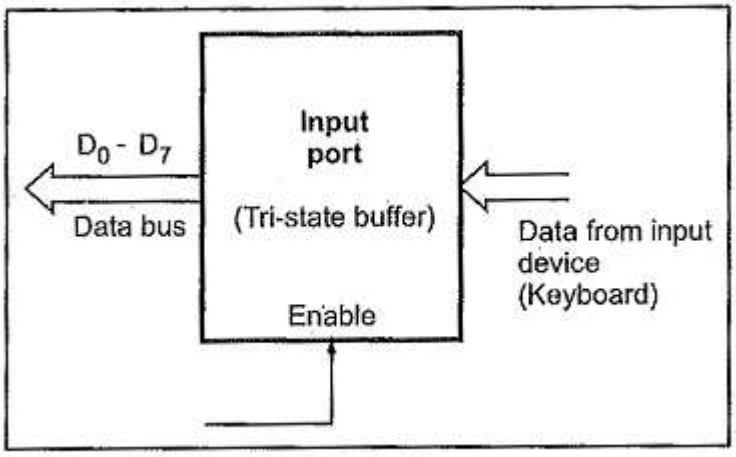


Fig. 4.28

Output Port:

It is used to send data to the output device such as display from the microprocessor. The simplest form of output port is a latch. The output device is connected to the microprocessor through latch as shown in the Fig. 4.29.

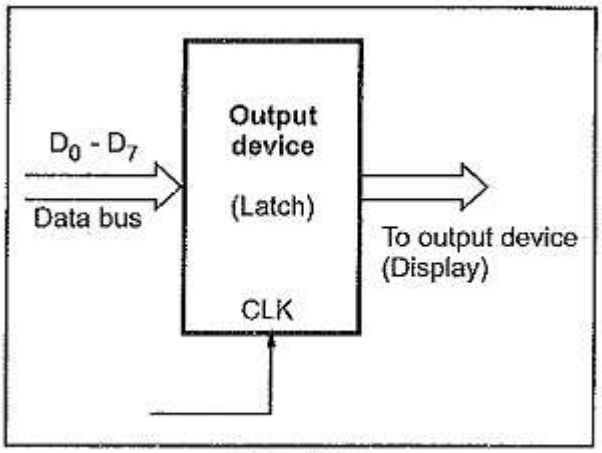


Fig. 4.29

When microprocessor wants to send data to the output device, it puts the data on the data bus and activates the clock signal of the latch, latching the data from the data bus at the output of latch. It is then available at the output of latch for the output device.

Input Output Interfacing Techniques:

The most of the microprocessor's support isolated I/O system. It partitions memory from I/O, via software, by having instructions that specifically access (address) memory, and others that specifically access I/O. When these instructions are decoded by the microprocessor, an appropriate control signal is generated to activate either memory or I/O operation. In 8085, IO/M signal is used for this purpose. The 8085 outputs a logic '1' on the IO/M line for an I/O operation and a logic '0' for memory, operation. In 8085, it is possible to connect 64 Kbyte memory and 256 I/O ports in the system since 8085 sends 16 bit address for memory and 8 bit address for I/O. I/O devices can be Input Output Interfacing Techniques to an 8085A system in two ways :

1. **I/O Mapped I/O**
2. **Memory mapped I/O**

In I/O mapped I/O, the 8085 uses IO/M signal to distinguish between I/O read/write and memory read/write operations. The 8085 has separate instructions IN and OUT for I/O data transfer. When 8085 executes IN or OUT instruction, it places device address (port number) on the demultiplexed low order address bus as well as the high order address bus. In other words, we can say that higher order address bus duplicates the contents of demultiplexed low-order address bus, when 8085 microprocessor executes an IN or OUT instruction. For example, if the device address is 60H then the contents on A_{15} to A_0 will be as follows :

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0

Here, A_8 follows A_0, A_9 follows A_1 and so on, as shown below.

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Device Address
A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	
0	1	1	0	0	0	0	0	60H

The instruction IN inputs data from an input device (such as keyboard) into the accumulator and the instruction OUT sends the contents of the accumulator to an output device such as LED display. These are two byte instructions. The second byte of the instruction specifies the address or the port number of an I/O device. As it is a byte, the address or port number can be any of the 256 combinations of eight bits, from 00H to FFH. Therefore, the 8085 can communicate with 256 different I/O devices. When we want to Input Output Interfacing Techniques, it is necessary to assign a device address or a port number. Before going to see this device address logic, we will examine how the 8085 executes IN and OUT instructions.

- The objective of Interfacing I/O peripherals:
  -is to obtain information or results from process.

  -to store, process or display

- The instructions IN and OUT perform this operation.
- The following examples shows the process of instruction

The timing diagram against this instruction *'IN 80H'* execution is as follows

Address	Hex Codes	Mnemonic	Comment
6000	DB	IN 80H	Content from the port address 80H will be written back on to the Accumulator
6001	80		80H as port address

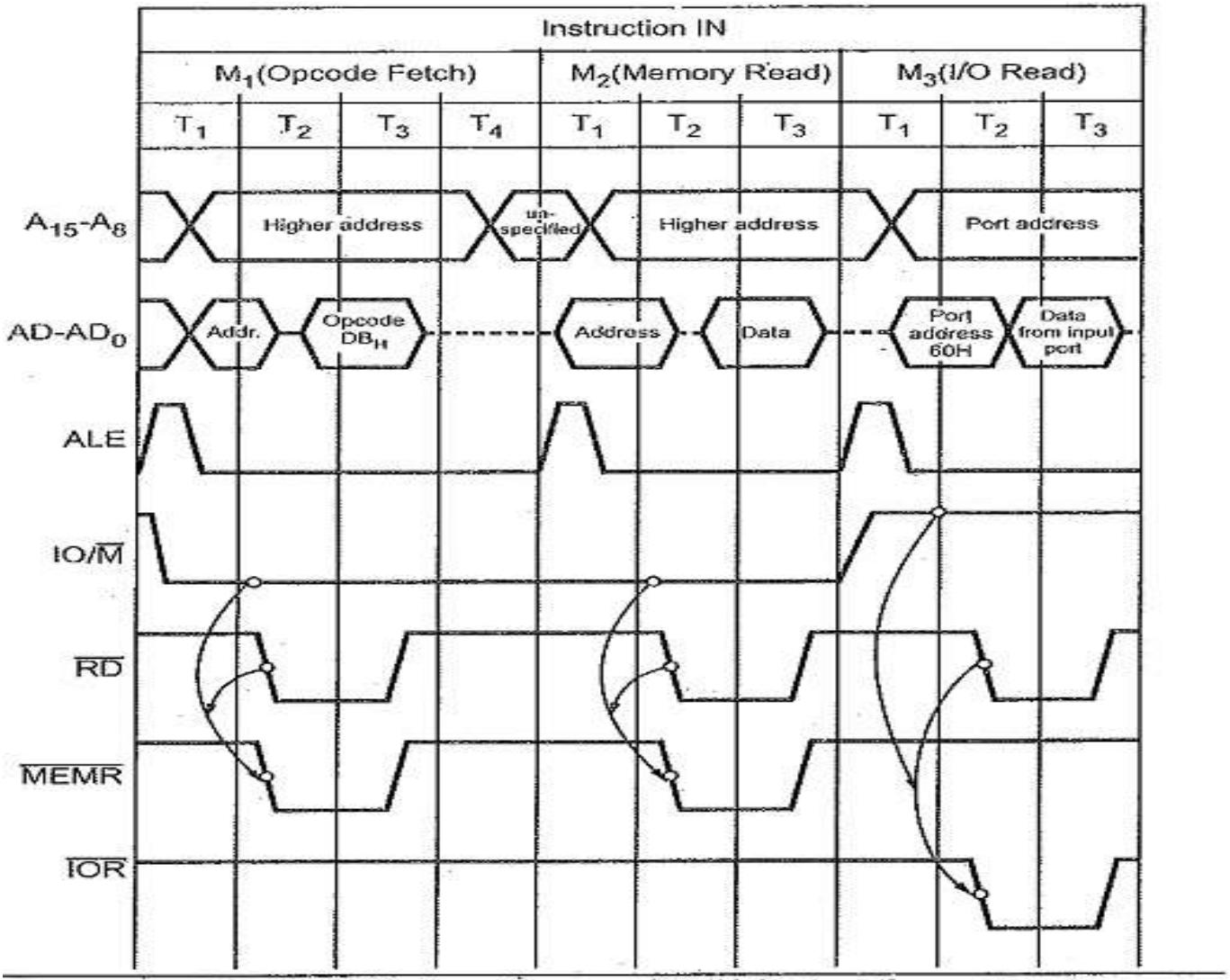


Fig. 4.30 Timing diagram for IN instruction

Fig. 4.30 shows the timing diagram of the IN instruction. It has three machine cycles. As usual, first cycle is opcode fetch machine cycle. The opcode fetch cycle is followed by one memory read machine cycle to read the address of port.

In the third machine cycle (I/O read) the 8085 microprocessor places the address of the input port on the low-order address bus AD_7-AD_0 as well as on the high-order address hubs A_{15}-A_8 and asserts the RD signal. During T_2 and T_3 of the machine cycle, RD and IO/M signals are 0 and 1 respectively, which activates IOR signal. The IOR signal enables the input port arid the data from the input port is placed on the data bus and transferred into the accumulator.

The timing diagram against this instruction '*OUT 81H*' execution is as follows

Address	Hex Codes	Mnemonic	Comment
6050	D3	OUT 81H	Content from the Accumulator will be moved into the port address 81H
6051	81		81H as port address

Fig. 4.31 shows the timing diagram of OUT instruction. It has, three machine cycles. The first machine cycle is an opcode fetch machine cycle, which reads the opcode of OUT instruction from the memory. The second machine cycle is a memory read machine cycle. This machine cycle reads the address of the port. In the third machine cycle (I/O write), the 8085 microprocessor places this address of the output port on the low-order address bus as well as on the high order address bus and asserts the WR signal. During T_2 and T_3 of the I/O write machine cycle, WR and ION signals are 0 and 1 respectively, which activates IOW signal. The IOW signal enables the output port and the data from the accumulator is sent to the output port.

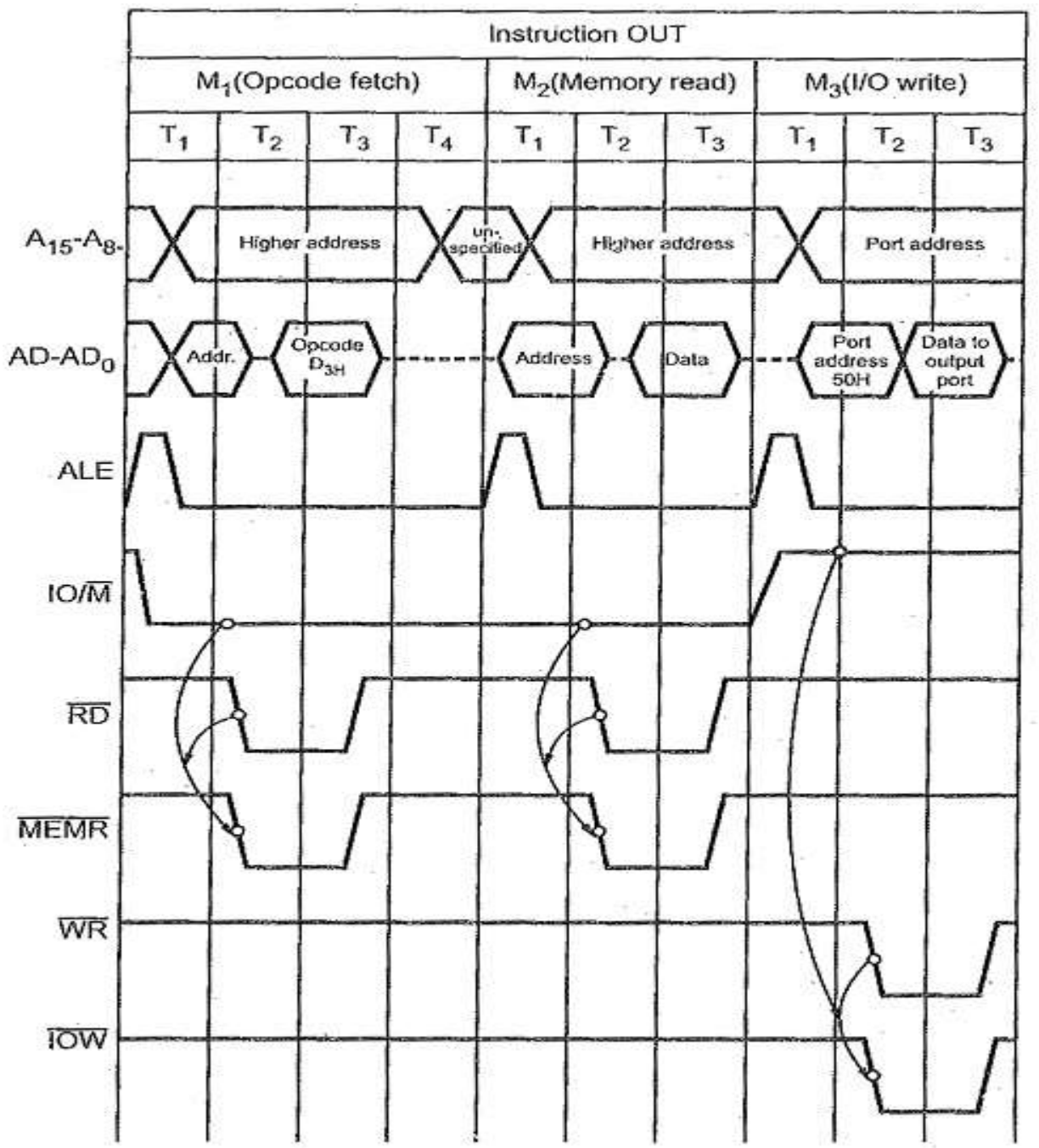


Fig. 4.31 Timing Diagram for OUT Instruction

UNIT-IV

I/O Peripherals Interface

• Figure 2 shows one of the way to decode address lines to obtain output address 01 H

. • The line address A 7 -A 0 is connected to eight NAND gates function as decoder.

• The line A 0 is connected directly, A 7 -A 1 are connected through inverter gates.

• The G 2 gate is combined with gate G 1 and IO/W control signal to generate select signal I/O whenever both signals are active low.

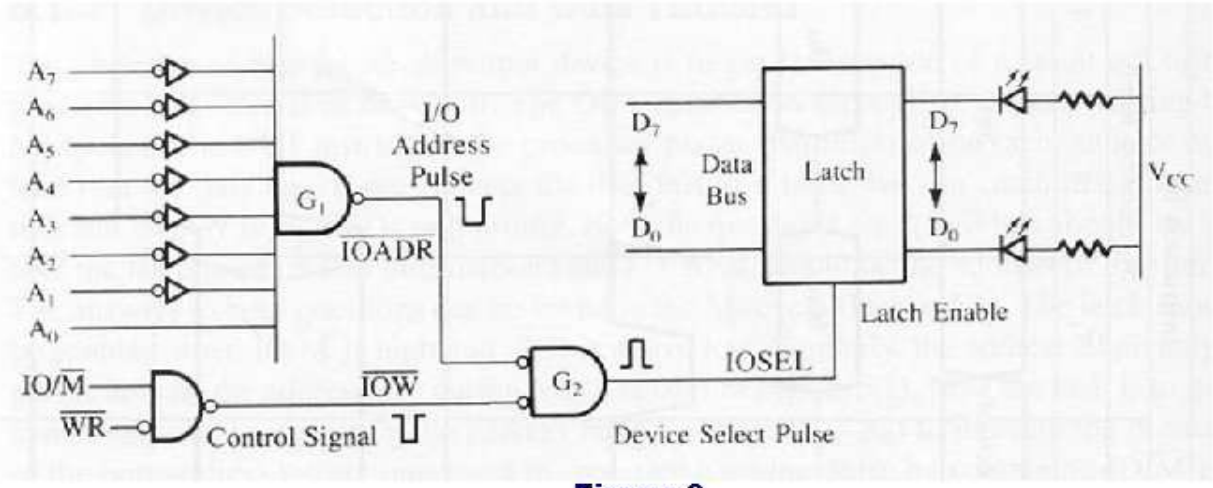


Figure 2

Note: In IO interfacing, only one segment of the address bus (low or high addresses) is sufficient (both segment have same address).

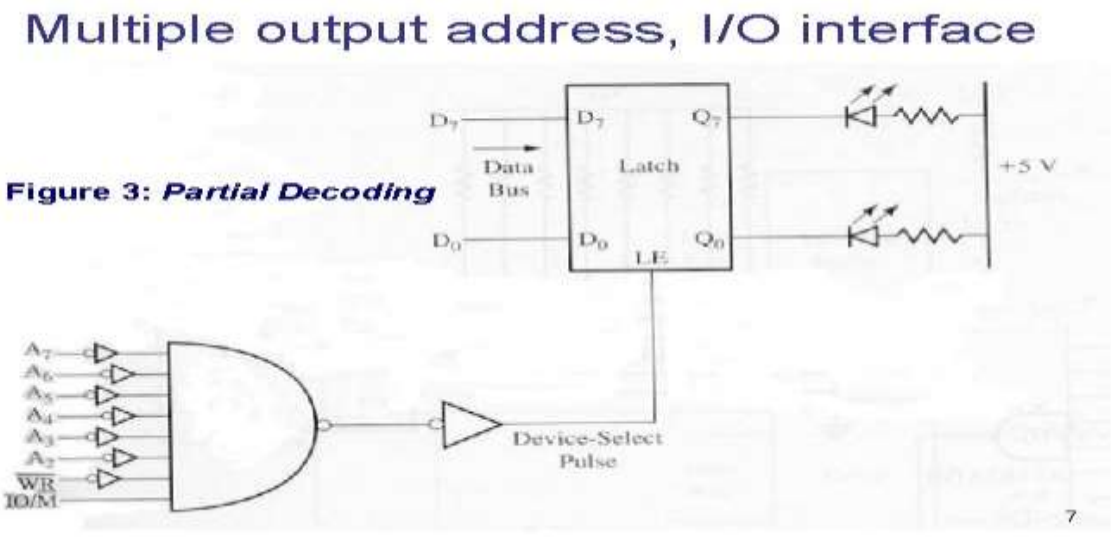
4

Multiple output address, I/O interface (Partial decoder):

• Figure 2 have shown the technique to decode I/O output address in **absolute** decoding technique.

• There is another technique which is cost saving - partial decoding.

• This technique gives the flexibility to user as to use more than one addresses to one output or input device.



Multiple output address, I/O interface Figure 3: Partial Decoding

• The address lines A 1 and A 0 are unused.

• Depending on the logic input given at address lines A 1 and A 0;

– the output addresses: 00 h, 01 h, 02 h or 03 h, which refer to the same output device.

• The multiple address is normally used in a small system;

– OK if those addresses are not being used by any other system, input or output devices.

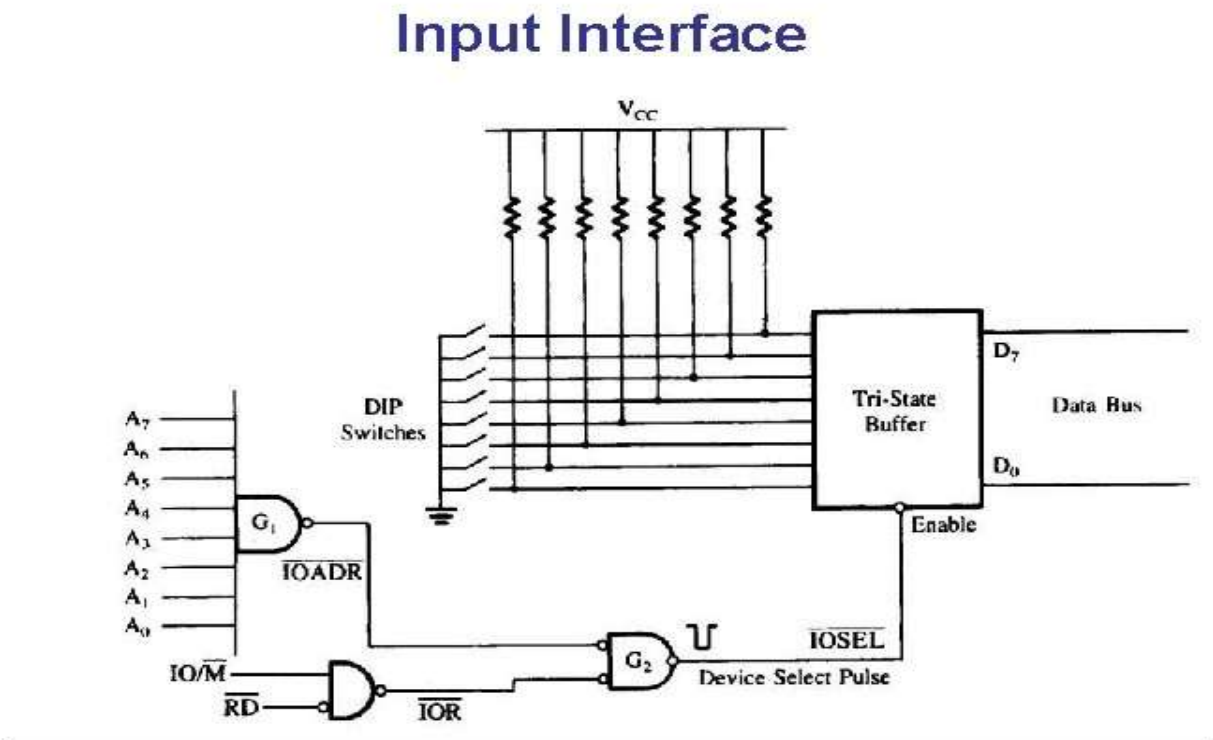
Input Interface

• The assembly instruction for that circuit in fig. 4 is IN FFH.

• Note: FFH = 11111111 binary • The line address is decoded using NAND gates.

• When address A 7 -A 0 is active high (FFH), the output of NAND gate will have an active low signal and then combined with control signals IOR at G 2.

• Suppose the µp run the IN FFH instruction, data at DIP switches will be placed at data bus and copied to accumulator.



Input Interface Figure 4: IN FFH

I/O interface using decoder

• Circuit in figure 5 decode input and output device at once using 3 -to-8 decoder and four input NAND gates.

• The address lines A 2, A 1 and A 0 are used as inputs to decoder, and the remaining line address A 7 ke A 3 is used to enable the decoder chip.

• The decoder has eight outputs;

therefore, we can use the decoder to address eight kinds of input and output devices.

I/O interface using decoder

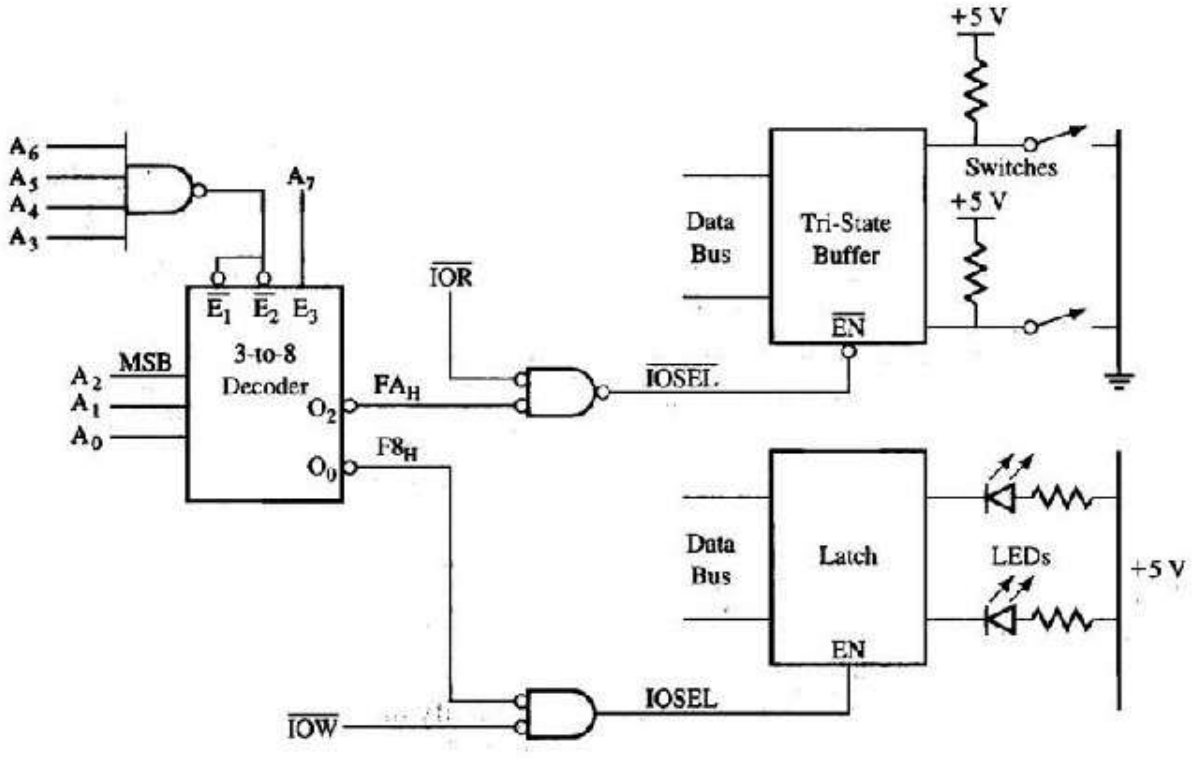
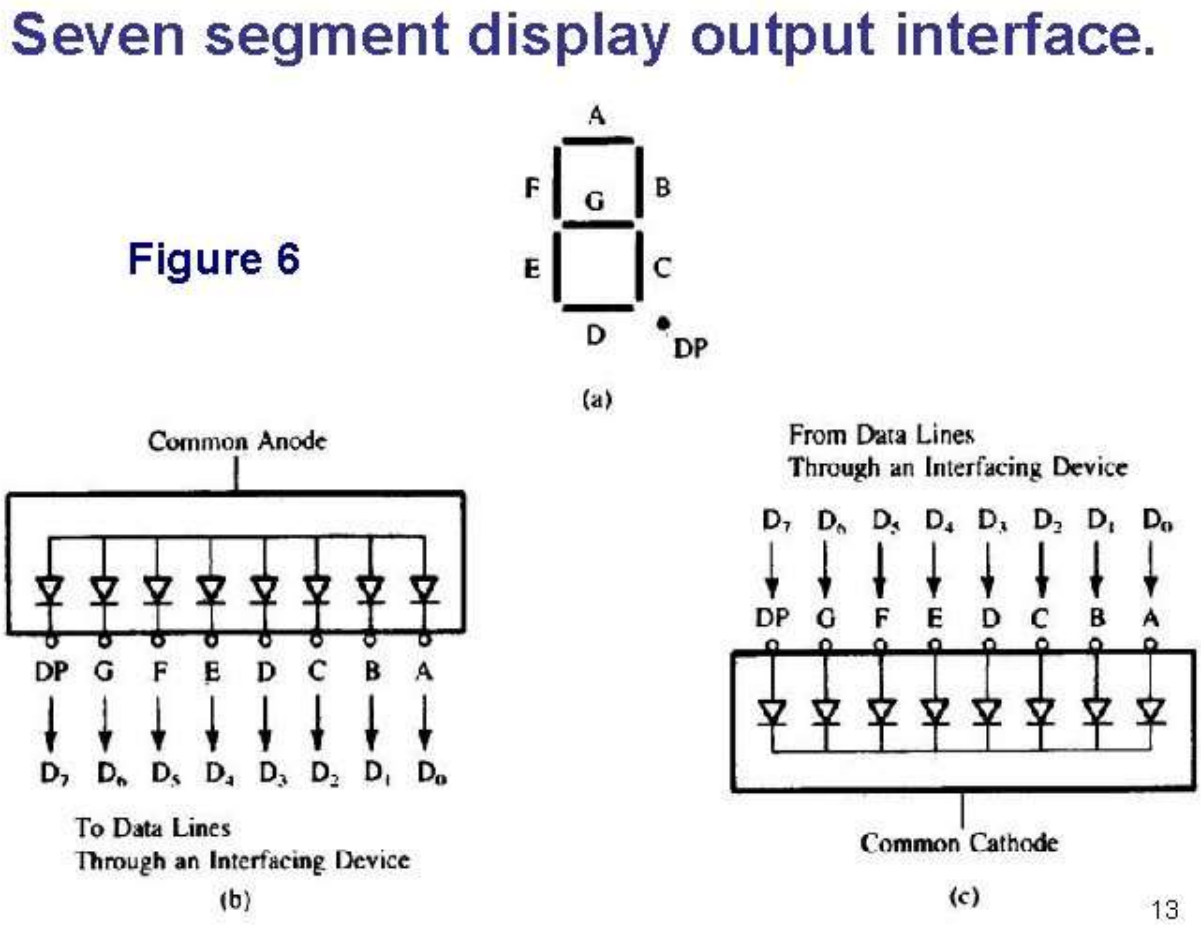


Figure 5

11

I/O interface using decoder Figure 5



Seven segment display output interface. Figure 6

Seven segment display output interface

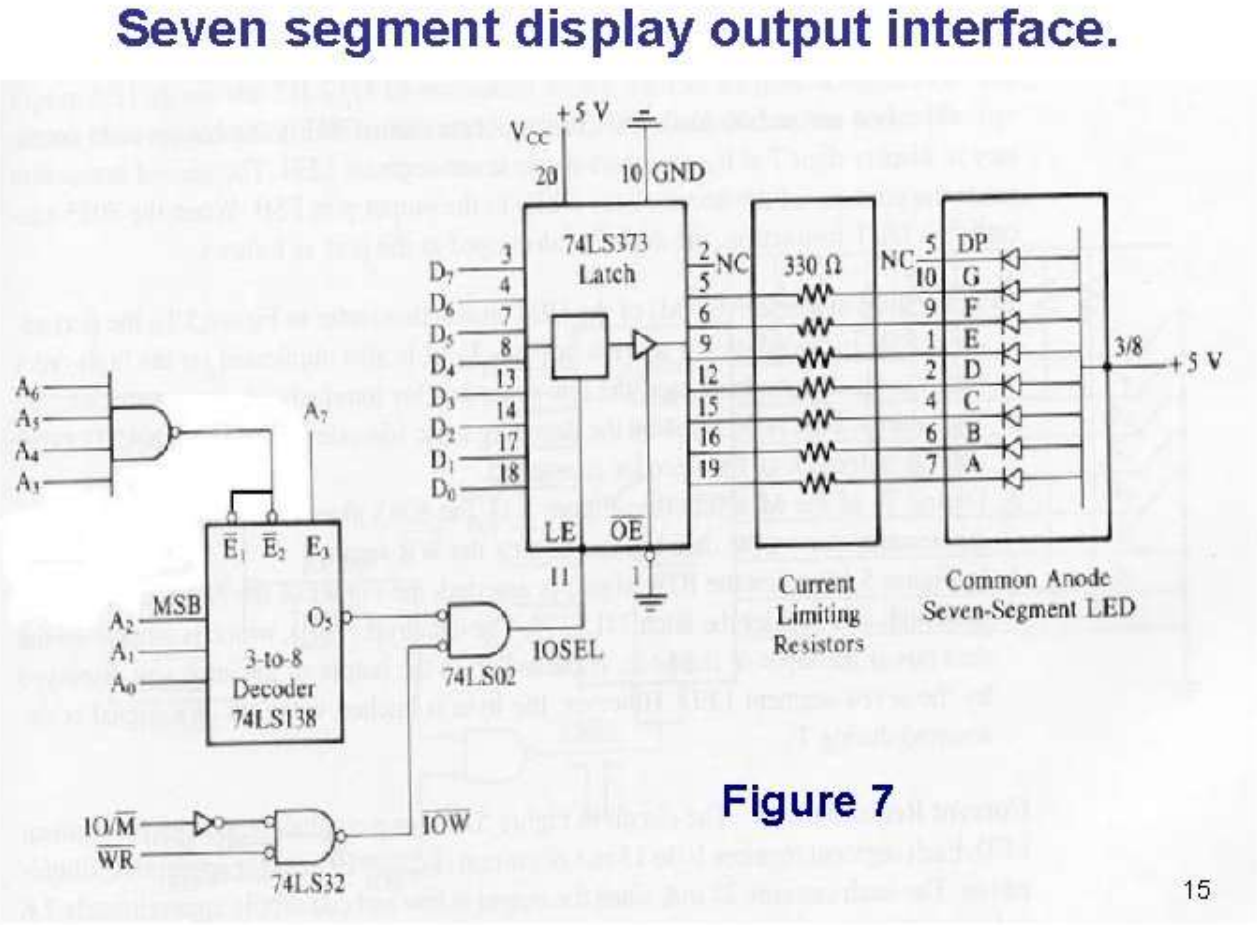
• Fig 6 shows two different type of 7 -segment display; common cathode and common anode.

• 7 -segment display consists of a few LEDs and are arranged physically as

shown in figure 7 a.

• It has seven segment from A to G that normally connected to data bus D 0 to D 6 respectively.

• If decimal point is used, D 7 will be connected to DP; and left unconnected if it is unused.



Seven segment display output interface. Figure 7

Seven segment display output interface.

• Fig. 7 shows the example to interface seven segment display and address decoder with an address of FDH.

• The common anode display is used therefore 0 logic is needed to activate the segment. • Suppose to display number 4 at seven segment display, therefore the segment F, G, B and C have to be activated

. • Follows are the instructions to execute it: –

MVI A, 66 H

OUT FDH

Data lines: D 7 D 6 D 5 D 4 D 3 D 2 D 1 D 0

Segments: NC G F E D C B A

Bits: X 1 1 0 0 1 1 0 = 66 H

Absolute Vs Partial Decoding:

Full Address Decoding	Partial Address decoding
1. All higher address lines are decoded to select the memory or I/O device.	1. Few higher address lines are decoded to select the memory or I/O device.
2. More hardware is required to design decoding logic.	2. Hardware required to design decoding logic is less and sometimes it can be eliminated.
3. Higher cost for decoding circuit.	3. Less cost for decoding circuit.
4. No Multiple addresses.	4. It has a advantage of multiple addresses.
5. Used in large systems	5. Used in small systems

UNIT-IV

Interrupts in 8085

Interrupts are the signals generated by the external devices to request the microprocessor to perform a task. There are 5 interrupt signals, i.e. TRAP, RST 7.5, RST 6.5, RST 5.5, and INTR.

Interrupt Service Routine (ISR)

A small program or a routine that when executed, services the corresponding interrupting source is called an ISR.

Interrupt are classified into following groups based on their parameter –

- Vector interrupt – In this type of interrupt, the interrupt address is known to the processor. For example: RST7.5, RST6.5, RST5.5, TRAP.

INTERRUPT	VECTOR ADDRESS
TRAP (RST 4.5)	24 H
RST 5.5	2C H
RST 6.5	34 H
RST 7.5	3C H

- Non-Vector interrupt – In this type of interrupt, the interrupt address is not known to the processor so, the interrupt address needs to be sent externally by the device to perform interrupts. For example: INTR.
- Maskable interrupt – In this type of interrupt, we can disable the interrupt by writing some instructions into the program. For example: RST7.5, RST6.5, RST5.5.
- Non-Maskable interrupt – In this type of interrupt, we cannot disable the interrupt by writing some instructions into the program. For example: TRAP.
- Software interrupt – In this type of interrupt, the programmer has to add the instructions into the program to execute the interrupt. There are 8 software interrupts in 8085, i.e. RST0, RST1, RST2, RST3, RST4, RST5, RST6, and RST7.
- Hardware interrupt – There are 5 interrupt pins in 8085 used as hardware interrupts, i.e. TRAP, RST7.5, RST6.5, RST5.5, INTR.

TRAP

It is a non-maskable interrupt, having the highest priority among all interrupts. By default, it is enabled until it gets acknowledged. In case of failure, it executes as ISR and sends the data to backup memory. This interrupt transfers the control to the location 0024H.

RST7.5

It is a maskable interrupt, having the second highest priority among all interrupts. When this interrupt is executed, the processor saves the content of the PC register into the stack and branches to 003CH address.

RST 6.5

It is a maskable interrupt, having the third highest priority among all interrupts. When this interrupt is executed, the processor saves the content of the PC register into the stack and branches to 0034H address.

RST 5.5

It is a maskable interrupt. When this interrupt is executed, the processor saves the content of the PC register into the stack and branches to 002CH address.

INTR

It is a maskable interrupt, having the lowest priority among all interrupts. It can be disabled by resetting the microprocessor.

When INTR signal goes high, the following events can occur –

- The microprocessor checks the status of INTR signal during the execution of each instruction.
- When the INTR signal is high, then the microprocessor completes its current instruction and sends active low interrupt acknowledge signal.
- When instructions are received, then the microprocessor saves the address of the next instruction on stack and executes the received instruction.

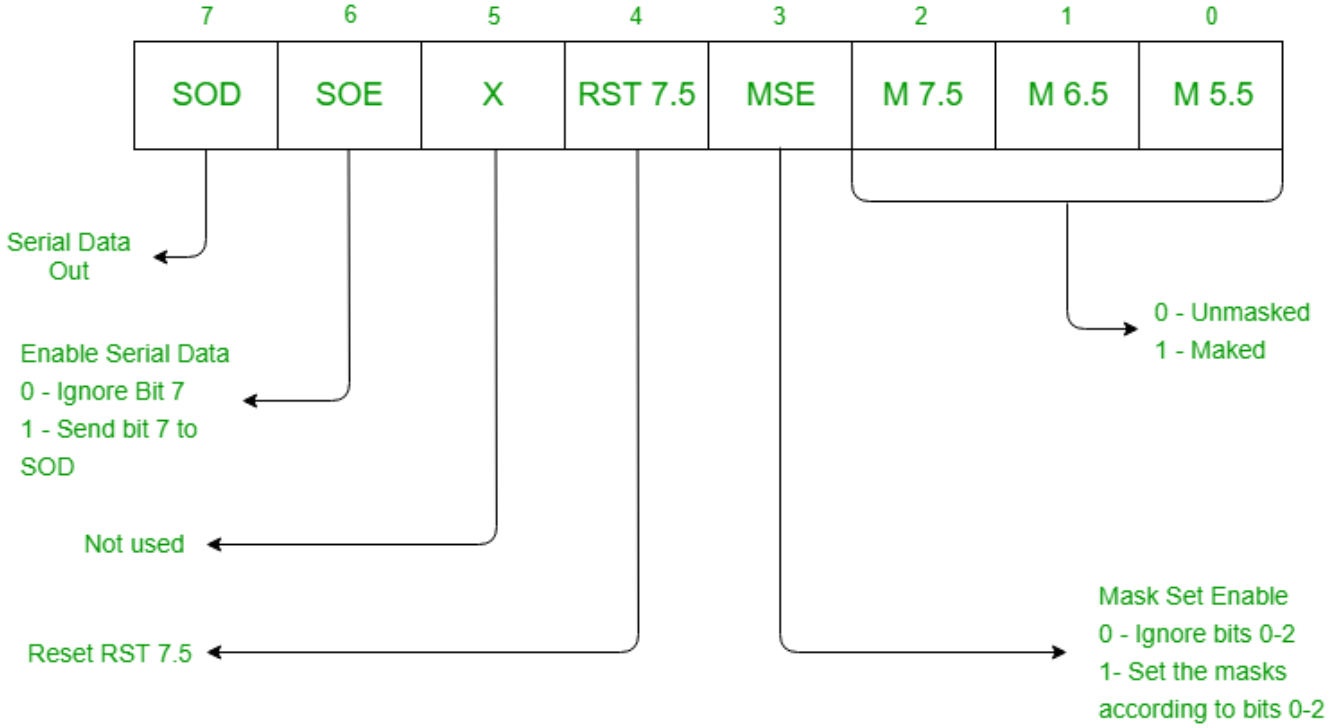
Note – INTA is not an interrupt, it is used by the microprocessor for sending acknowledgement. TRAP has the highest priority, then RST7.5 and so on.

Priority of Interrupts –

When microprocessor receives multiple interrupt requests simultaneously, it will execute the interrupt service request (ISR) according to the priority of the interrupts.

SIM and RIM instructions in 8085

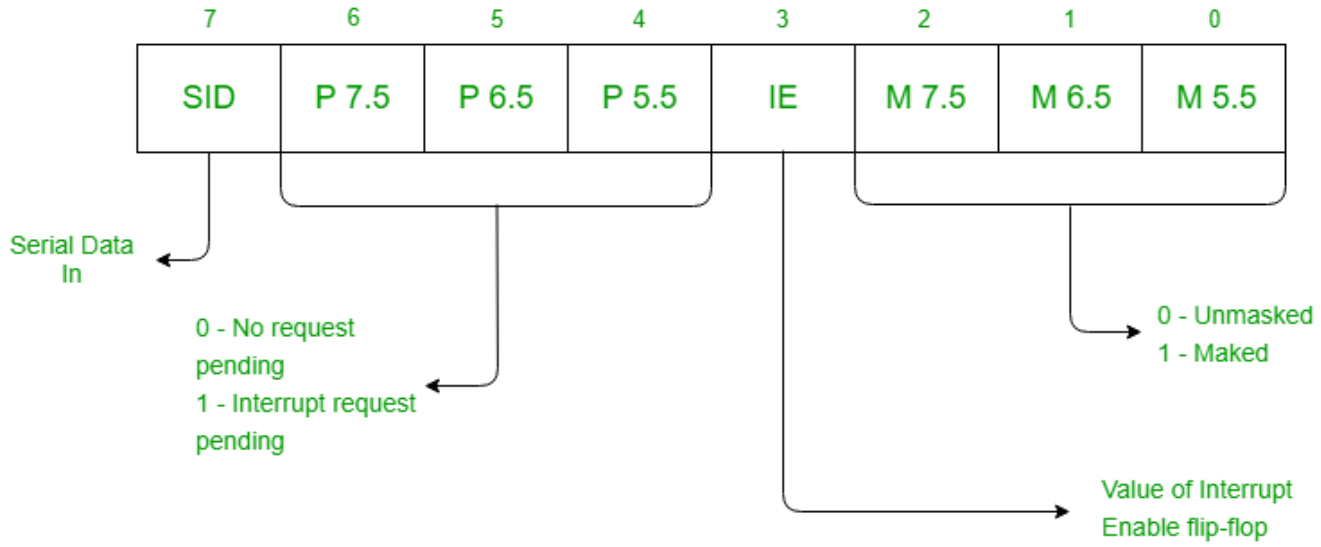
Set Interrupt Mask (SIM) :
In 8085 Instruction set, SIM stands for “Set Interrupt Mask”. It is 1-Byte instruction and it is a multi-purpose instruction.



The main uses of SIM instruction are –

- Masking/unmasking of RST7.5, RST6.5, and RST5.5
- Reset to 0 RST7.5 flip-flop.
- Perform serial output of data.

Read Interrupt Mask (RIM):
In 8085 Instruction set, RIM stands for “Read Interrupt Mask”. It is a 1-Byte multi-purpose instruction.



It is used for the following purposes –

- To check whether RST7.5, RST6.5, and RST5.5 are masked or not.
- To check whether interrupts are enabled or not.
- To check whether RST7.5, RST6.5, or RST5.5 interrupts are pending or not.
- To perform serial input of data.

UNIT-4

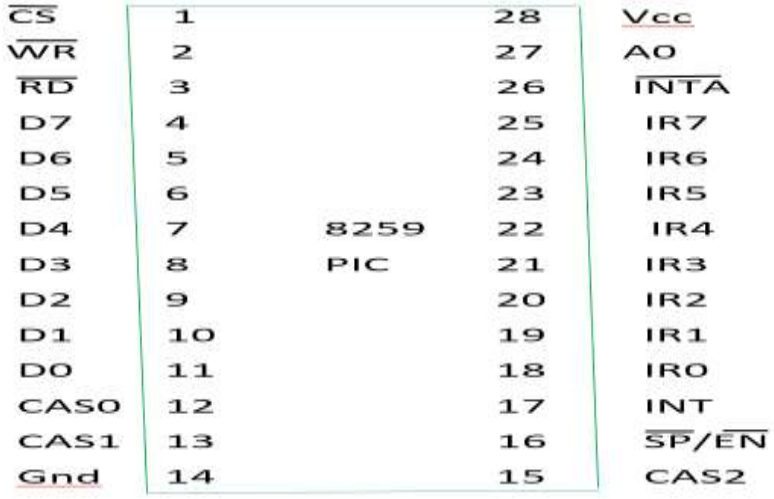
Programmable Interrupt Controller (8259)

8259 microprocessor is defined as Programmable Interrupt Controller (PIC) microprocessor. There are 5 hardware interrupts and 2 hardware interrupts in 8085 and 8086 respectively. But by connecting 8259, we can increase the interrupt handling capability. 8259 combines the multi interrupt input sources into a single interrupt output. Interfacing of single PIC provides 8 interrupts inputs from IR0-IR7.
For example, Interfacing of 8085 and 8259 increases the interrupt handling capability of 8085 microprocessor from 5 to 8 interrupt levels.

Features of 8259 PIC microprocessor –

1. Intel 8259 is designed for Intel 8085 and Intel 8086 microprocessor.
2. It can be programmed either in level triggered or in edge triggered interrupt level.
3. We can mask individual bits of interrupt request register.
4. We can increase interrupt handling capability upto 64 interrupt level by cascading further 8259 PIC.
5. Clock cycle is not required.

Pin Diagram of 8259 –

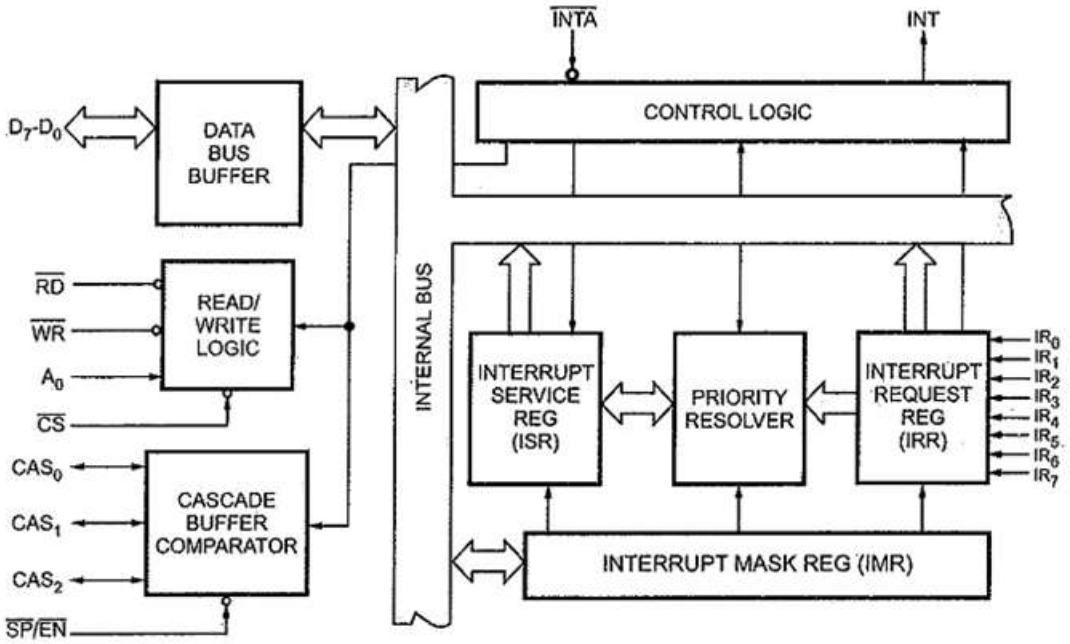


Pin Diagram Description of 8259

Symbol	Pin No.	Type	Name and function of Pin
Vcc	28	Input(I)	**Supply** : +5V supply
GND	14	i	Ground
CS*	1	I	**Chip Select :** A low on this pin enables RD* and WR* communication between the CPU and the 8259A. INTA functions are independent of CS.
WR*	2	I	**Write:** A low on this pin when CS is low enables the 8259A to accept command words from the CPU.
RD*	3	I	**Read :**When the CS pin is low, a low signal on this pin allows the 8259A to transfer status information onto the data bus for the CPU.
D7-D0	4-11	I/O	**Bidirectional data bus :** Information related to control, status, and interrupt vectors flows through this bus.
CAS0-CAS2	12,13,15	I/O	**Cascade lines:** The CAS lines from a private 8259A bus to control a multiple 8259A structure. These pins are outputs for a master 8259A and input for a slave 8259A.
SP*/EN*	16	I/O	**Slave program/Enable buffer:** This is a dual function pin. When in the buffered mode

Symbol	Pin No.	Type	Name and function of Pin
			it can be used as an output to control buffer transceivers(EN).When not in the buffered mode it is used as an input to designate a master(SP =1) or Slave(SP = 0).
INT	17	O	**Interrupt:** This pin goes high whenever a valid interrupt request is asserted. It is used to interrupt the CPU, thus it is connected to the CPU's interrupt pin.
IR0-IR7	18-25	I	**Interrupt Request :** Asynchronous inputs. An interrupt request is executed raising an IR input(low to high), and holding it high until it is acknowledged(Edge triggered mode), or just by a high level on an IR input(Level Triggered Mode).
INTA*	26	I	**Interrupt Acknowledge:** This pin is used to enable 8259A interrupt-vector data onto the data bus by a sequence of interrupt acknowledge pulses issued by the CPU.
A0	27	I	**AO Address line:** This pin acts in conjunction with the CS*, WR* and RD* pins. It is used by the 8259 to decipher various command words the CPU writes and status the CPU weighs to read. It's usually linked to the CPU's A0 address line (A1 for 8086, 8088).

Block Diagram of 8259 PIC microprocessor –



Block diagram of 8259A

The Block Diagram consists of 8 blocks which are – Data Bus Buffer, Read/Write Logic, Cascade Buffer Comparator, Control Logic, Priority Resolver and 3 registers- ISR, IRR, IMR.

1. **Data bus buffer –** This Block is used as a mediator between 8259 and 8085/8086 microprocessor by acting as a buffer. It takes the control word from the 8085 microprocessor and transfer it to the control logic of 8259 microprocessor. After selection of Interrupt by 8259 microprocessor (based on priority of the interrupt), it transfer the opcode of the selected Interrupt and address of the Interrupt service sub routine to the other connected microprocessor. The data bus buffer consists of 8 bits represented as D0-D7 in the block diagram. Thus, shows that a maximum of 8 bits data can be transferred at a time.
2. **Read/Write logic –** This block works only when the value of pin CS is low (as this pin is active low). This block is responsible for the flow of data depending upon the inputs of RD and WR. These two pins are active low pins used for read and write operations.

3. **Control logic –** It is the center of the PIC and controls the functioning of every block. It has pin INTR which is connected with other microprocessor for taking interrupt request and pin INT for giving the output. If 8259 is enabled, and the other microprocessor Interrupt flag is high then this causes the value of the output INT pin high and in this way 8259 responds to the request made by other microprocessor.
4. **Interrupt request register (IRR) –** It stores all the interrupt level which are requesting for Interrupt services.
5. **Interrupt service register (ISR) –** It stores the interrupt level which are currently being executed.
6. **Interrupt mask register (IMR) –** It stores the interrupt level which have to be masked by storing the masking bits of the interrupt level.
7. **Priority resolver –** It examines all the three registers and set the priority of interrupts and according to the priority of the interrupts, interrupt with highest priority is set in ISR register. Also, it reset the interrupt level which is already been serviced in IRR.
8. **Cascade buffer –** To increase the Interrupt handling capability, we can further cascade more number of pins by using cascade buffer. So, during increment of interrupt capability, CSA lines are used to control multiple interrupt structure.

SP/EN (Slave program/Enable buffer) pin is when set to high, works in master mode else in slave mode. In Non Buffered mode, SP/EN pin is used to specify whether 8259 work as master or slave and in Buffered mode, SP/EN pin is used as an output to enable data bus

The events occur as follows in an MCS-80/85 system:

1. One or more of the INTERRUPT REQUEST lines (IR7 – 0) are raised high, setting the corresponding IRR bit(s).

2. The 8259A evaluates these requests, and sends an INT to the CPU, if appropriate.

3. The CPU acknowledges the INT and responds with an INTA pulse.

4. Upon receiving an INTA from the CPU group, the highest priority ISR bit is set, and the corresponding IRR bit is reset. The 8259A will also release a CALL instruction code (11001101) onto the 8-bit Data Bus through its D7 – 0

pins.6.

5. This CALL instruction will initiate two more INTA pulses to be sent to the 8259A from the CPU group.

6. These two INTA pulses allow the 8259A to release its preprogrammed subroutine address onto the Data Bus. The lower 8-bit address is released at the first INTA pulse and the higher 8-bit address is released at the second INTA pulse.

7. This completes the 3-byte CALL instruction released by the 8259A. In the AEOI mode the ISR bit is reset at the end of the third INTA pulse. Otherwise, the ISR bit remains set until an appropriate EOI command is issued at the end of the interrupt sequence.

UNIT-4

DMA -DMA controller (8257)

DMA stands for Direct Memory Access. It is designed by Intel to transfer data at the fastest rate. It allows the device to transfer the data directly to/from memory without any interference of the CPU.

Using a DMA controller, the device requests the CPU to hold its data, address and control bus, so the device is free to transfer data directly to/from the memory. The DMA data transfer is initiated only after receiving HLDA signal from the CPU.

How DMA Operations are Performed?

Following is the sequence of operations performed by a DMA −

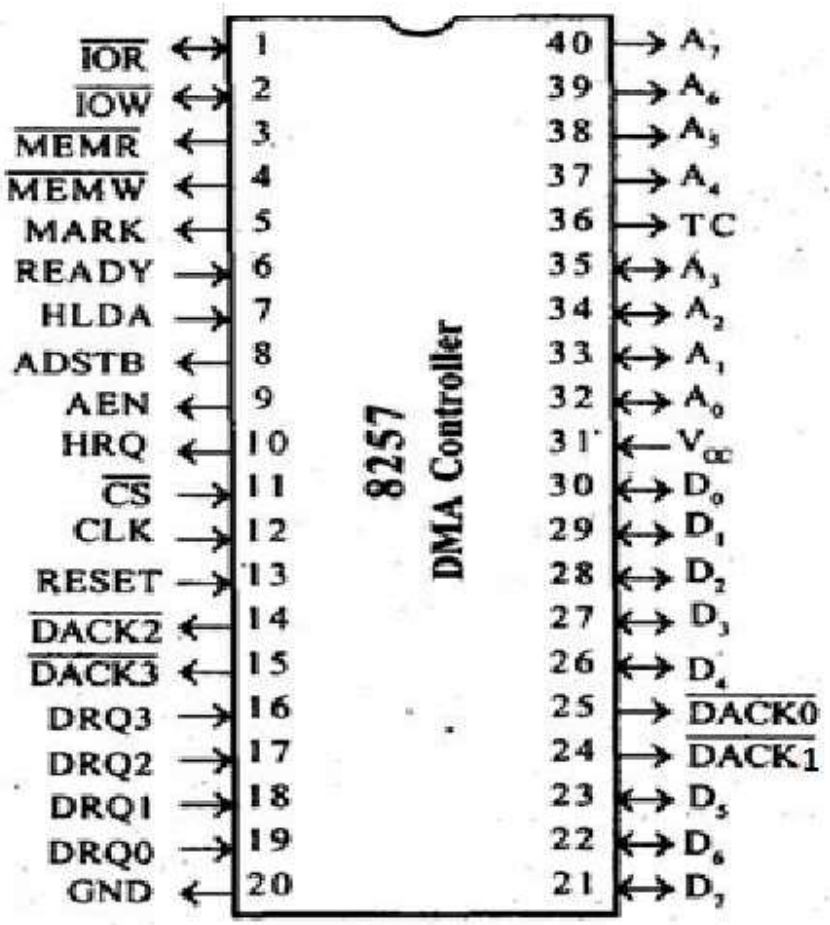
- Initially, when any I/O device has to send data between the device and the memory, the device has to send DMA request (DRQ) to DMA controller.
- The DMA controller sends Hold request (HRQ) to the CPU and waits for the CPU to assert the HLDA.
- Then the microprocessor tri-states all the data bus, address bus, and control bus. The CPU leaves the control over bus and acknowledges the HOLD request through HLDA signal.
- Now the CPU is in HOLD state and the DMA controller has to manage the operations over buses between the CPU, memory, and I/O devices.

Features of 8257

- It has four channels which can be used over four I/O devices.
- Each channel has 16-bit address and 14-bit counter.
- Each channel can transfer data up to 64kB.
- Each channel can be programmed independently.
- Each channel can perform read transfer, write transfer and verify transfer operations.
- It generates MARK signal to the peripheral device that 128 bytes have been transferred.
- It requires a single-phase clock.
- Its frequency ranges from 250Hz to 3MHz.
- It operates in 2 modes, i.e., Master mode and Slave mode.

8257 Pin Description

The following image shows the pin diagram of a 8257 DMA controller –



DRQ_0–DRQ3

These are the four individual channel DMA request inputs, which are used by the peripheral devices for using DMA services. When the fixed priority mode is selected, then DRQ_0 has the highest priority and DRQ_3 has the lowest priority among them.

$DACK_o$ – $DACK_3$

These are the active-low DMA acknowledge lines, which updates the requesting peripheral about the status of their request by the CPU. These lines can also act as strobe lines for the requesting devices.

D_o – D_7

These are bidirectional, data lines which are used to interface the system bus with the internal data bus of DMA controller. In the Slave mode, it carries command words to 8257 and status word from 8257. In the master mode, these lines are used to send higher byte of the generated address to the latch. This address is further latched using ADSTB signal.

IOR

It is an active-low bidirectional tri-state input line, which is used by the CPU to read internal registers of 8257 in the Slave mode. In the master mode, it is used to read data from the peripheral devices during a memory write cycle.

IOW

It is an active low bi-direction tri-state line, which is used to load the contents of the data bus to the 8-bit mode register or upper/lower byte of a 16-bit DMA address register or terminal count register. In the master mode, it is used to load the data to the peripheral devices during DMA memory read cycle.

CLK

It is a clock frequency signal which is required for the internal operation of 8257.

RESET

This signal is used to RESET the DMA controller by disabling all the DMA channels.

A_o - A_3

These are the four least significant address lines. In the slave mode, they act as an input, which selects one of the registers to be read or written. In the master mode, they are the four least significant memory address output lines generated by 8257.

CS

It is an active-low chip select line. In the Slave mode, it enables the read/write operations to/from 8257. In the master mode, it disables the read/write operations to/from 8257.

A_4 - A_7

These are the higher nibble of the lower byte address generated by DMA in the master mode.

READY

It is an active-high asynchronous input signal, which makes DMA ready by inserting wait states.

HRQ

This signal is used to receive the hold request signal from the output device. In the slave mode, it is connected with a DRQ input line 8257. In Master mode, it is connected with HOLD input of the CPU.

HLDA

It is the hold acknowledgement signal which indicates the DMA controller that the bus has been granted to the requesting peripheral by the CPU when it is set to 1.

MEMR

It is the low memory read signal, which is used to read the data from the addressed memory locations during DMA read cycles.

MEMW

It is the active-low three state signal which is used to write the data to the addressed memory location during DMA write operation.

ADST

This signal is used to convert the higher byte of the memory address generated by the DMA controller into the latches.

AEN

This signal is used to disable the address bus/data bus.

TC

It stands for 'Terminal Count', which indicates the present DMA cycle to the present peripheral devices.

MARK

The mark will be activated after each 128 cycles or integral multiples of it from the beginning. It indicates the current DMA cycle is the 128th cycle since the previous MARK output to the selected peripheral device.

V_{cc}

It is the power signal which is required for the operation of the circuit.

INTERNAL ARCHITECTURE OF 8257:

The functional Block Diagram of DMA controller(8257) is shown in Figure 3.8.2 and thedescription are as follows: It consists of five functional blocks:

a) Data bus buffer

b) Control logic

c) Read/write logic

d) Priority Resolver

e) DMA channels

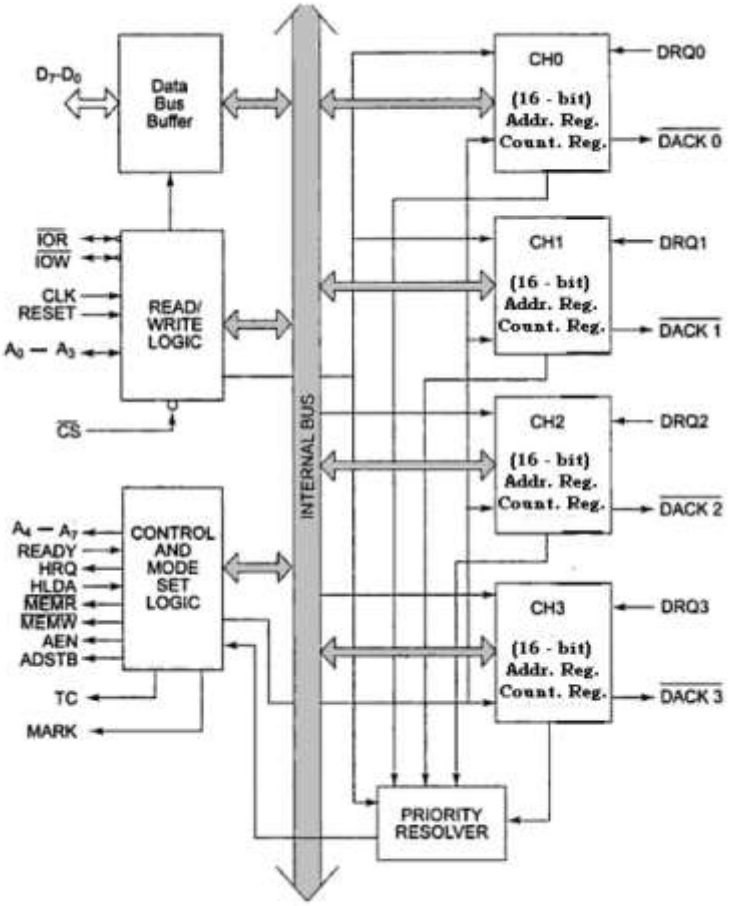


Figure 3.8.2 Functional Block Diagram of 8257

Data Bus Buffer:

8-bit Tristate, bidirectional buffer interfaces the internal bus of 8257 withthe external system bus under the control of various control signals.

Read/Write Logic:

In the slave mode, the read/write logic accepts the I/O Read or I/O Write signals, decodes the Ao-A3 lines and either writes the contents of the data bus to the addressed internal register or reads the selected register depending upon whether IOW or IOR signal is activated. In master mode, the read/write logic generates the IOR and IOW signals to control the dataflow to or from the selected peripheral.

Control Logic:

The control logic controls the sequences of operations and generates the required control signals like AEN, ADSTB, MEMR, MEMW, TC and MARK along with the address lines A4-A7, in master mode.

Priority Resolver:

The priority resolver resolves the priority of the four DMA channels dependingupon whether normal priority or rotating priority is programmed.

Register Organization of 8257:

The 8257 performs DMA operation over four independent DMA channels with the following Registers.

1. DMA Address Register

Each DMA channel has one DMA address register. The function of this register is to store the address of the starting memory location, which will

be accessed by the DMA channel. The device that wants to transfer data over a DMA channel, will access the block of the memory with the starting address stored in the DMA Address Register.

2. Terminal Count Registers

Each of the four DMA channels of 8257 has one terminal count register (TC). This 16-bit registers used for ascertaining that the data transfer through a DMA channel cease or stops after the required number of DMA cycles.

After each DMA cycle, the terminal count register content will be decremented by one and finally it becomes zero after the required number of DMA cycles are over. The bits 14 and 15 of this register indicate the type of theDMA operation (transfer).

3. Mode Set Register

The mode set register is used for programming the 8257 as per the requirements of the system. The function of the mode set register is to enable the DMA channels individually and also to set the various modes of operation as shown in Figure 3.8.3.

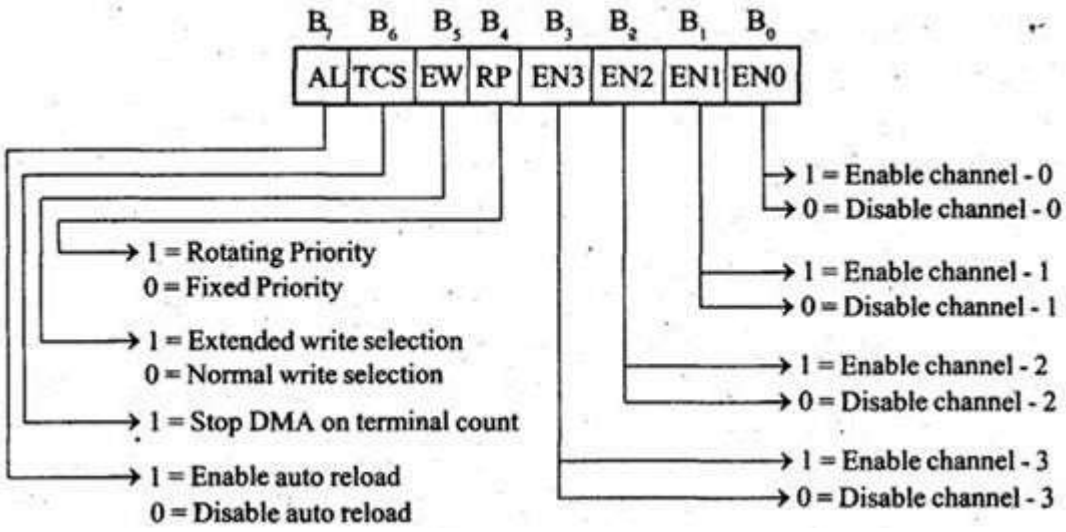


Figure 3.8.3 Mode Set Register

The bits Do-D3 enable one of the four DMA channels of 8257.If the TC STOP bit is set, the selected channel is disabled after the terminal count condition is reached, and it further prevents any DMA cycle on the channel. If the TC STOP bit is programmed to be zero, the channel is not disabled, even after the count reaches zero and further request are allowed on the same channel. The auto load bit, if set, enables channel 2 for the repeat block chaining operations, without immediate software intervention between the two successive blocks. The extended write bit, if set to '1', extends the duration of MEMW and IOW signals by activating them earlier, which is useful in interfacing the peripherals with different access times.

4. Status register

The lower order 4-bits of this register contain the terminal count status for the four individual channels. If any of these bits is set, it indicates that the specific channel has reached the terminal count condition. The update flag is not affected by the read operation. This flag can only be cleared by resetting 8257. The update flag is set every time, the channel 2 registers are loaded with contents of the channel 3 registers. It is cleared by the completion of the first DMA cycle of the new block. This register can only read.

UNIT-4

DIGITAL-TO-ANALOG (D/A) CONVERTERS

Basic Concepts

Figure 13.1(a) shows a block diagram of a 3-bit D/A converter; it has three digital input lines (D2, D1, and Do) and one output line for the analog signal. The three input lines can assume eight ($2^3 = 8$) input combinations from 000 to 111, D_2 being the most significant bit (MSB) and D_o being the least significant bit (LSB). If the input ranges from 0 to 1 V, it can be divided into eight equal parts (8 V); each successive input is 8 V higher than the previous combination, as shown in Figure 13.1(b).
The following points can be summarized from the graph:

1. The 3-bit D/A converter has eight possible combinations. 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. In this example, the LSB =1/ 8 V.
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.

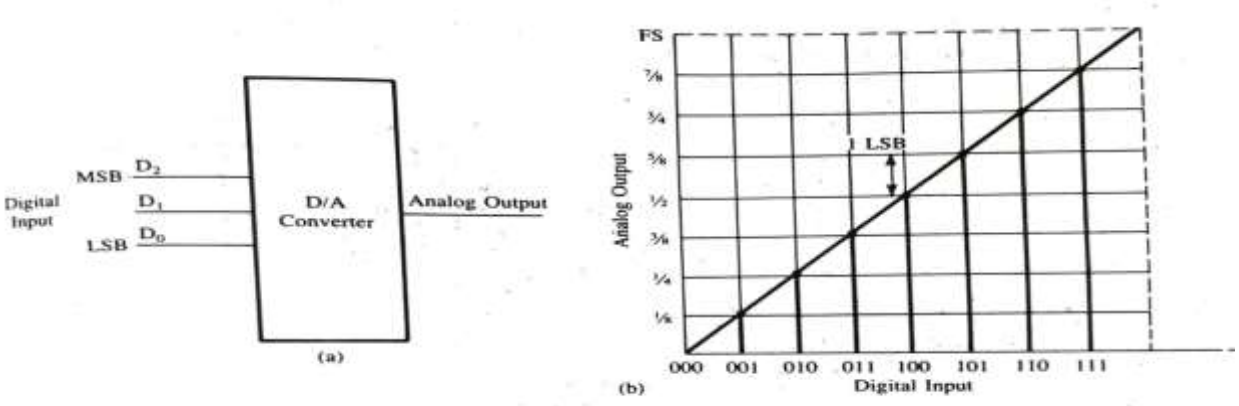


FIGURE 13.1
A 3-Bit D/A Converter: Block Diagram (a) and Digital Input vs. Analog Output (b)

R/2R LADDER NETWORK

The R/2R ladder network shown in Figure 13.4 uses only two resistor values. The resistors are connected in such a way that for any number of inputs, the total current I_T is in binary proportion. The R/2R ladder network is used commonly in designing integrated D/A converters. The interfacing of 8-bit integrated D/A converters is illustrated.

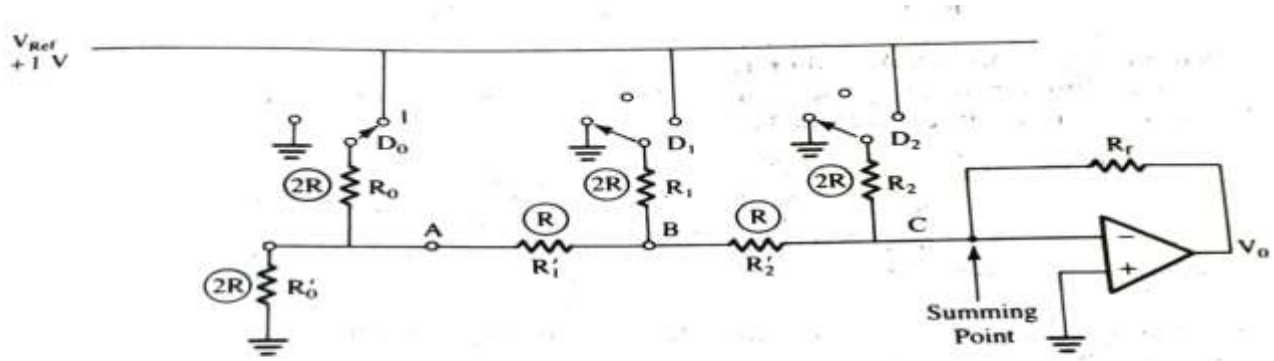


FIGURE 13.4
R/2R Ladder Network

Illustration: Interfacing an 8-Bit D/A Converter with the 8085

HARDWARE DESCRIPTION

The circuit shown in Figure 13.5(a) includes an 8-input NAND gate and a NOR gate (negative AND) as the address decoding logic, the 74LS373 as a latch, and an industry- standard 1408 D/A converter. The address lines A7-Ao are decoded using the 8-input NAND gate, and the output of the NAND gate is combined with the control signal $\overline{IOW}$. When the microprocessor sends the address FFH, the output of the negative AND gate enables the latch, and the data bits are placed on the input lines of the converter for conversion.

The 1408 has eight input data lines A1 (MSB) through A8 (LSB); the convention of labeling MSB to LSB is opposite to that of what is normally used for the data bus in the microprocessor. It requires 2 mA reference current for full-scale input and two power supplies Vcc = +5 V and VEE =-15 V (VEE can range from -5 V to -15 V).

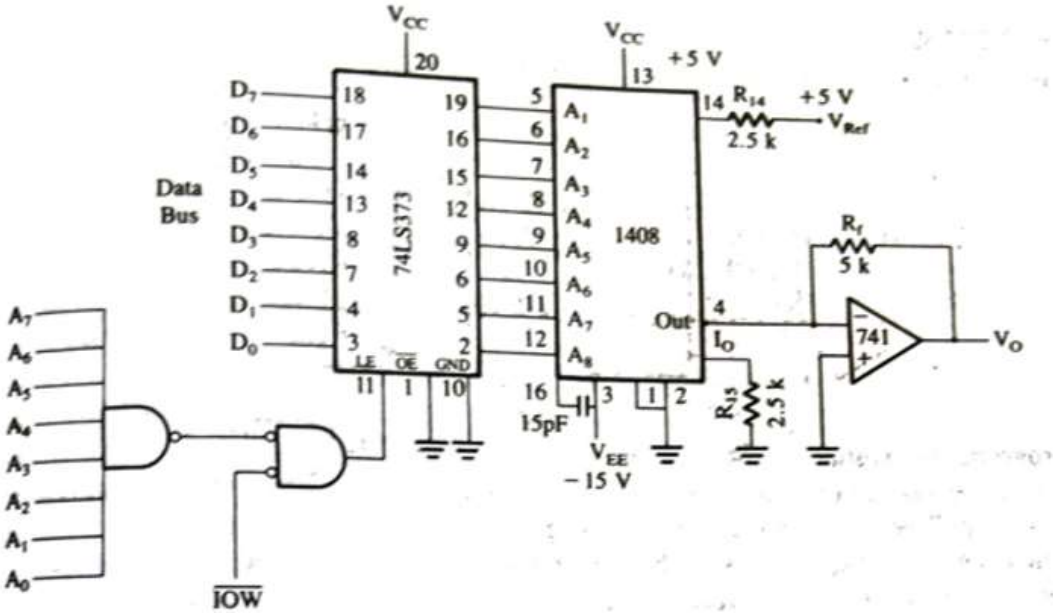


Figure 13.5 Interfacing the 1408 D/A Converter

The total reference current source is determined by the resistor R14 and the voltage VRef. The resistor R15 is generally equal to R14 to match the input impedance of the reference source. The output Io is calculated as follows:

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

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

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

$$I_O = \frac{5\text{ V}}{2.5\text{ k}}\left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} + \frac{1}{64} + \frac{1}{128} + \frac{1}{256}\right)$$
$$= 2\text{ mA } (255/256)$$
$$= 1.992\text{ mA}$$

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

$$V_O = 2\text{ mA } (255/256) \times 5\text{ k}$$
$$= 9.961\text{ V}$$

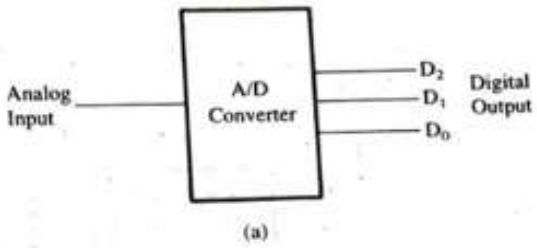
UNIT-4

ANALOG-TO-DIGITAL (A/D) CONVERTERS

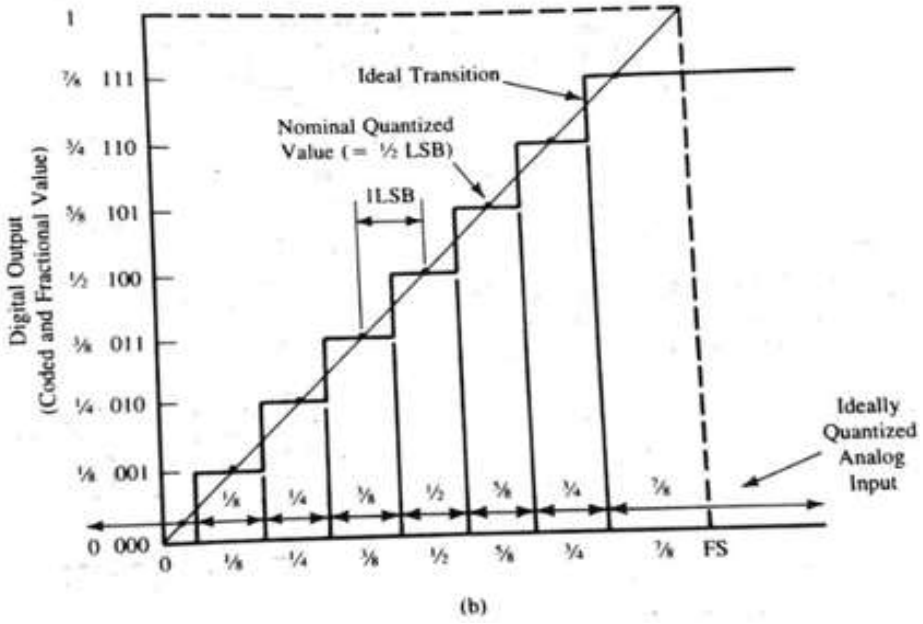
The A/D conversion is a quantizing process whereby an analog signal is represented by equivalent binary states; this is opposite to the D/A conversion process. The most commonly used A/D converters-the successive-approximation.

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)



(b)

Basic Concepts

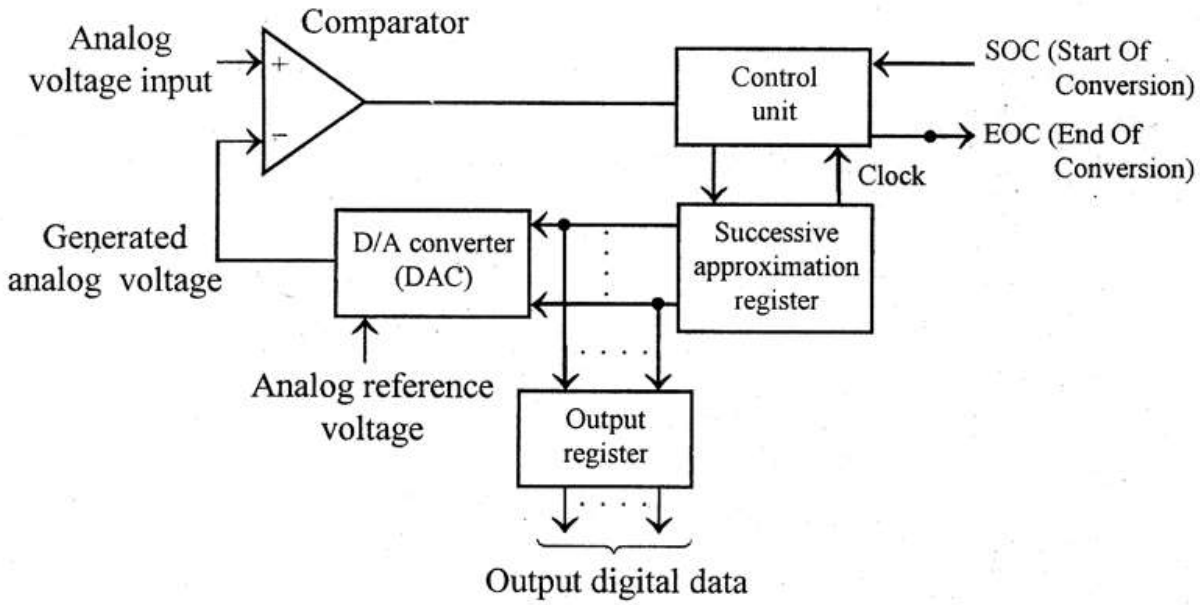
Figure 13.8(a) shows a block diagram of a 3-bit A/D converter. It has one input line for an analog signal and three output lines for digital signals. Figure 13.8(b) shows the graph of the analog input voltage (0 to 1 V) and the corresponding digital output signal. It shows eight (23) discrete output states from 000_2 to 111_2, each step being 1/8 V apart. This is defined as the resolution of the converter. The LSB, the MSB, and the full-scale output are calculated the same way as in D/A converters.

Successive-Approximation A/D Converter

Figure 13.9(a) shows the block diagram of a successive approximation A/D converter. It includes three major elements: the D/A converter, the successive approximation register (SAR), and the comparator. The conversion technique involves comparing the output of the D*/A* converter Vo with the analog input signal Vin. The digital input to the DAC is generated using the successive-approximation method (explained below). When the DAC output matches the analog signal, the input to the DAC is the equivalent digital signal.
The successive-approximation method of generating input to the DAC is similar to weighing an unknown material (e.g., less than 1 gram) on a chemical balance with a set of such fractional weights as 2 g, 4 g, 8 g, etc. The weighing procedure begins with the heaviest weight (1/2 g), and subsequent weights (in decreasing order) are added until the balance is tipped. The weight that tips the balance is removed, and the process is continued until the smallest weight is used. In the case of a 4-bit A/D converter, bit D3 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, is larger than the analog signal, bit D3 is turned off in the SAR and bit D2 is turned on. The process continues until the input reaches bit Do. Figure 13.9(b) illustrates a 4-bit conversion process. When bit D3 is turned on, the output exceeds the analog signal and, therefore, bit D3 is turned off. When the next three successive bits are turned on, the output becomes approximately equal to the analog signal.

The successive-approximation conversion process can be accomplished through either the software or hardware approach. In the software approach, an A/D converter is designed using a D/A converter, and the microprocessor plays the role of the counter and the SAR. For the hardware approach, a complete ADC, including a tri-state buffer, is now available as an integrated circuit on a chip. The

interfacing of both types of A/D converters is illustrated in the next section.



Interfacing 8-Bit A/D Converters

As an integrated circuit, the A/D converter includes all three elements-SAR, DAC, and comparator on a chip (Figure 13.10). In addition, it has a tri-state output buffer.

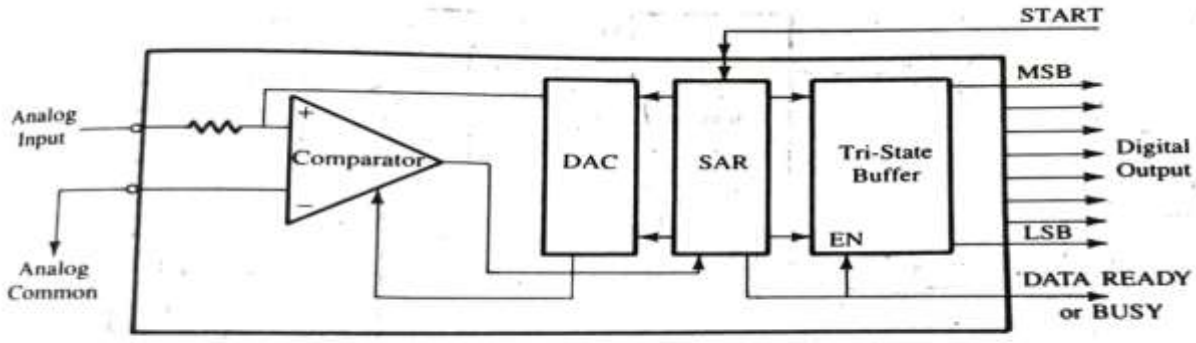


FIGURE 13.10
Block Diagram of a Typical Successive-Approximation A/D Converter as an Integrated Circuit

Typically, it has two control lines, START (or CONVERT) and DATA READY (or BUSY); they are TTL-compatible and can be active low or high depending upon the design.

A pulse to the START pin begins the conversion process and disables the tri-state output buffer. At the end of the conversion period, DATA READY becomes active and the digital output is made available at the output buffer. To interface an A/D converter with the microprocessor, the microprocessor should

1. send a pulse to the START pin. This can be derived from a control signal such as Write *(*WR).

2. wait until the end of the conversion. The end of the conversion period can be verified either by status checking (polling) or by using the interrupt.
3. read the digital signal at an input port.

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. The A/D converter has one input line for analog signal and eight output lines for converted digital signals. Typically, the analog signal can range from 0 to 10 V, or ±5 V. In addition, the converter shows two lines $\overline{\text{START}}$ and $\overline{\text{DR}}$ (Data Ready), both active low (the active logic level of these lines can be either low or high depending upon the design of a particular converter). When an active low pulse is sent to the $\overline{\text{START}}$ pin, the $\overline{\text{DR}}$ goes high and the output lines go into the high impedance state. The rising edge of the $\overline{\text{START}}$ pulse initiates the conversion. When the conversion is complete, the $\overline{\text{DR}}$ goes low, and the data are made available on the output lines that can be ready by the microprocessor. To interface this converter, we need one output port to send a $\overline{\text{START}}$ pulse and two input ports: one to check the status of the $\overline{\text{DR}}$ line and the other to read the output of the converter.

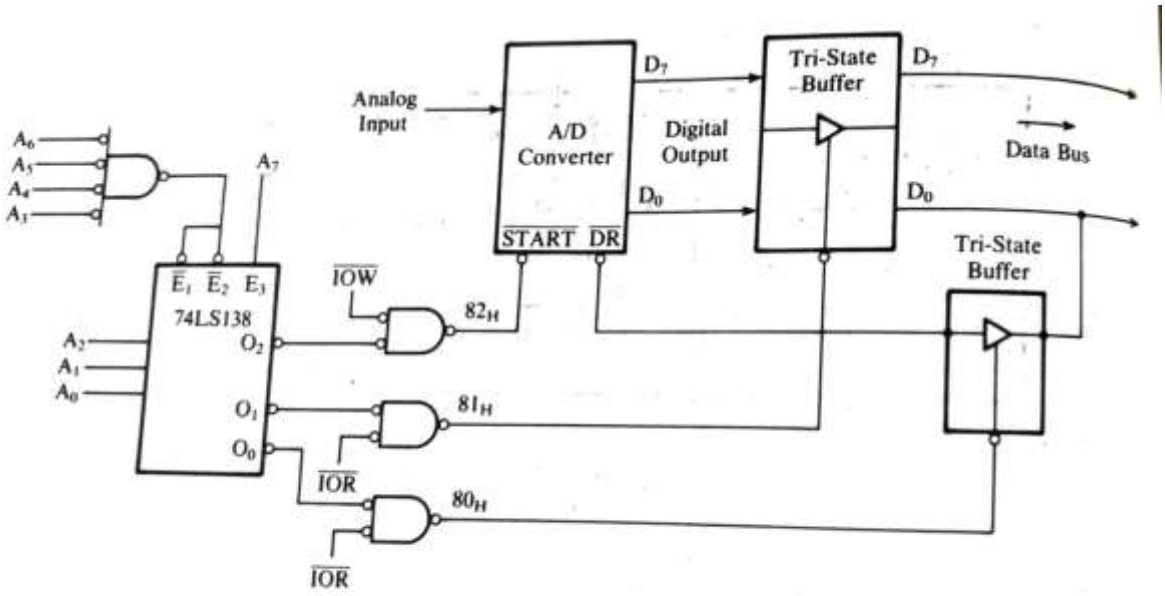


FIGURE 13.11
Interfacing an A/D Converter Using the Status Check

In Figure 13.11, the address decoding is performed by using the 3-to-8 decoder, (74LS138), the 4-input NAND gate, and inverters. Three output lines of the decoder are combined with appropriate control signals ($\overline{\text{IOW}}$ and $\overline{\text{IOR}}$) to assign three port addresses from 80H to 82H. The output port 82H is used to send a $\overline{\text{START}}$ pulse by writing the OUT instruction; in this case, we are interested in getting a pulse from the microprocessor, and the contents of the accumulator are irrelevant to start the conversion. However, for some converters the $\overline{\text{IOW}}$ pulse from the microprocessor may not be long enough to start the conversion. When the conversion begins, the $\overline{\text{DR}}$ (Data Ready) goes high and stays high until the conversion is completed. The status of the $\overline{\text{DR}}$ line is monitored by connecting the line to bit Do of the data bus through a tri-state buffer with the input port address 80H. The processor will continue to read port 80H until bit Do goes low. When the $\overline{\text{DR}}$ goes active, the data are available on the output lines of the converter, and the processor can access that data by reading the input port 81H.