UNIT-3: TIMING DIAGRAMS

3.1 Definitions:

Timing Diagram:

Timing Diagram is a graphical representation. It represents the execution time taken by each instruction in a graphical format. The execution time is represented in T-states.

Instruction Cycle:

The time required to execute an instruction is called instruction cycle.

or

The time taken by the processor to complete the execution of an instruction. An instruction cycle consists of one to six machine cycles.

Machine Cycle:

The time required to access the memory or input/output devices is called machine cycle.

or

The time required to complete one operation; accessing either the memory or I/O device. A machine cycle consists of three to six T-states.

T-State:

The machine cycle and instruction cycle takes multiple clock periods. A portion of an operation carried out in one system clock period is called as T-state.

or

Time corresponding to one clock period. It is the basic unit to calculate execution of instructions or programs in a processor.

Fetch cycle:

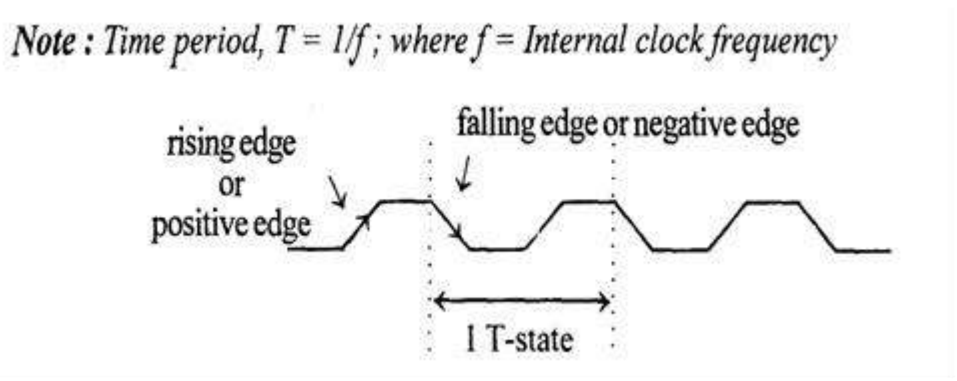
The fetch cycle in a microprocessor comprises of several time states during which the next instruction to be executed is copied (fetched) from the memory location (whose address is in the Program Counter) to the Instruction Register.

3.2 CONCEPT OF TIMING DIAGRAM:

The 8085 microprocessor has 5 (seven) basic machine cycles. They are

1. Opcode fetch cycle (4T)
2. Memory read cycle (3 T)
3. Memory write cycle (3 T)
4. I/O read cycle (3 T)

5. I/O write cycle (3 T)


Note : Time period, T = 1/f ; where f = Internal clock frequency
rising edge or positive edge
falling edge or negative edge
1 T-state

- Each instruction of the 8085 processor consists of one to five machine cycles, i.e., when the 8085 processor executes an instruction, it will execute some of the machine cycles in a specific order.

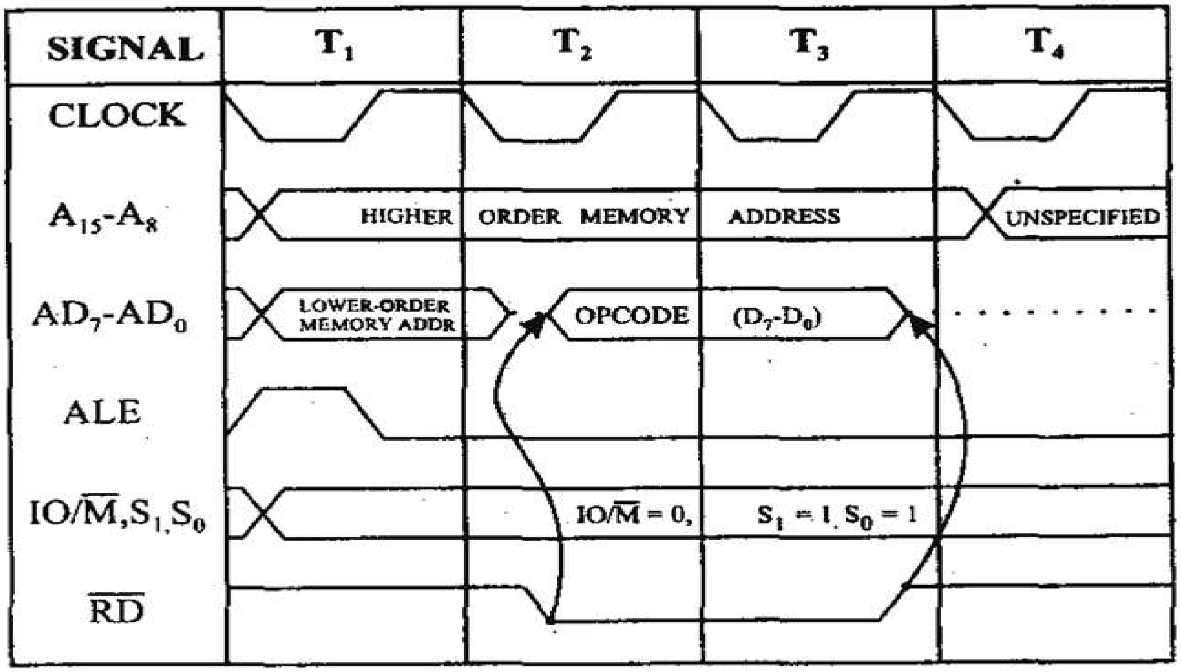
- The processor takes a definite time to execute the machine cycles. The time taken by the processor to execute a machine cycle is expressed in T-states.

- One T-state is equal to the time period of the internal clock signal of the processor.

- The T-state starts at the falling edge of a clock.

Opcode Fetch Machine Cycle:

- It is the first step in the execution of any instruction. The timing diagram of this cycle is given below.



- The following points explain the various operations that take place and the signals that are changed during the execution of opcode fetch machine cycle:

T1 clock cycle:

- The content of PC is placed in the address bus; AD0 - AD7 lines contains lower bit address and A8 – A15 contains higher bit address.
- **IO/M’** signal is low indicating that a memory location is being accessed. S1 and S0 also changed to the levels.
- ALE is high, indicates that multiplexed AD0 – AD7 act as lower order bus.

T2 clock cycle:

- Multiplexed address bus is now changed to data bus.
- The **(RD)’** signal is made low by the processor. This signal makes the memory device load the data bus with the contents of the location addressed by the processor.

T3 clock cycle:

- The opcode available on the data bus is read by the processor and moved to the instruction register.
- The **(RD)’** signal is deactivated by making it logic 1.

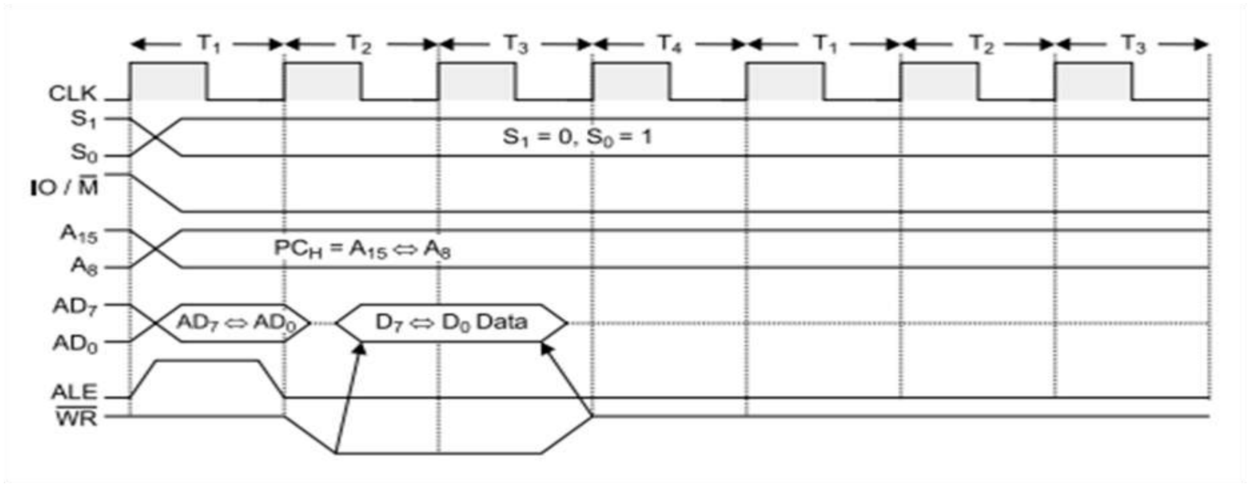
T4 clock cycle:

- The processor decode the instruction in the instruction register and generate the necessary control signals to execute the instruction. Based on the instruction further operations such as fetching, writing into memory etc. takes place.

3.3 DRAW TIMING DIAGRAM FOR MEMORY READ, MEMORY WRITE, I/O READ, I/O WRITE MACHINE CYCLE:

<u>Memory Read Machine Cycle:</u>

- The memory read cycle is executed by the processor to read a data byte from memory. The machine cycle is exactly same to opcode fetch except: a) It has three T-states b) The S0 signal is set to 0.



T1 state:

- The higher order address bus (A8-A15) and lower order address and data multiplexed (AD0-AD7) bus.
- ALE goes high so that the memory latches the (AD0-AD7) so that complete 16-bit address are available.
- The microprocessor identifies the memory read machine cycle from the status signals IO/M'=0, S1=1, S0=0. This condition indicates the memory read cycle.

T2 state:

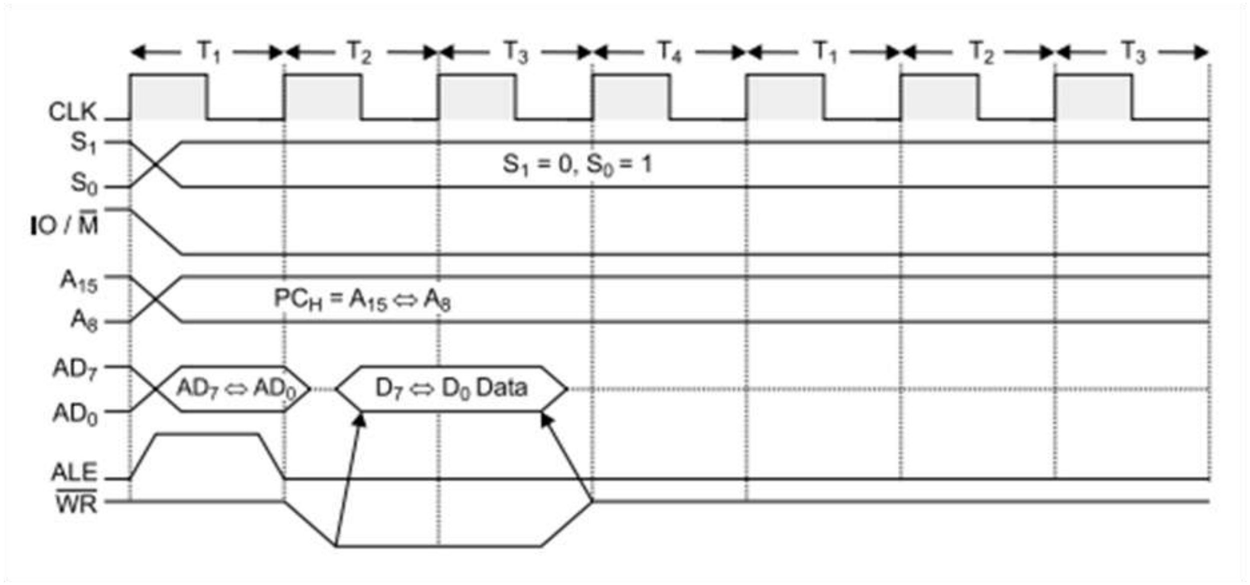
- Selected memory location is placed on the (D0-D7) of the A/D multiplexed bus. RD' goes LOW

T3 State:

- The data which was loaded on the previous state is transferred to the microprocessor.
- In the middle of the T3 state RD' goes high and disables the memory read operation.
- The data which was obtained from the memory is then decoded.

<u>Memory Write Machine Cycle:</u>

- The memory write cycle is executed by the processor to write a data byte in a memory location. The processor takes three T-states and (**WR)'**signal is made low.



T1 state:

- The higher order address bus (A8-A15) and lower order address and data multiplexed (AD0-AD7) bus.
- ALE goes high so that the memory latches the (AD0-AD7) so that complete 16-bit address are available.
- The microprocessor identifies the memory read machine cycle from the status signals IO/M'=0, S1=0, S0=1. This condition indicates the memory read cycle.

T2 state:

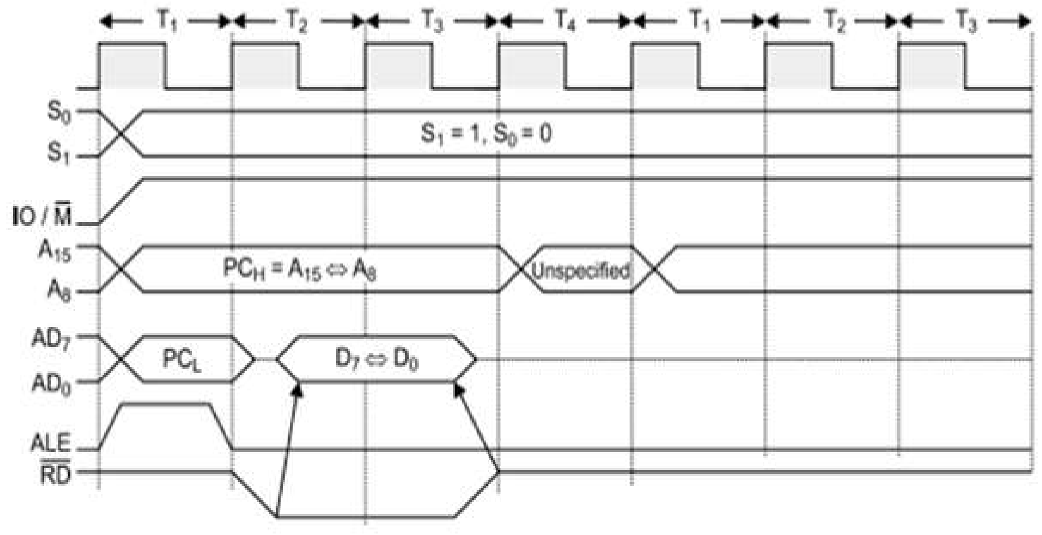
- Selected memory location is placed on the (D0-D7) of the A/D multiplexed bus. WR' goes LOW

T3 State:

- In the middle of the T3 state WR' goes high and disables the memory write operation. The data which was obtained from the memory is then decoded.

I/O Read Cycle:

The I/O read cycle is executed by the processor to read a data byte from I/O port or from peripheral, which is I/O mapped in the system. The 8-bit port address is placed both in the lower and higher order address bus. The processor takes three T-states to execute this machine cycle.



T1 state:

- The higher order address bus (A8-A15) and lower order address and data multiplexed (AD0-AD7) bus.
- ALE goes high so that the memory latches the (AD0-AD7) so that complete 16-bit address are available.
- The microprocessor identifies the I/O read machine cycle from the status signals IO/M'=1, S1=1, S0=0. This condition indicates the I/O read cycle.

T2 state:

- Selected memory location is placed on the (D0-D7) of the A/D multiplexed bus. RD' goes LOW

T3 State:

- The data which was loaded on the previous state is transferred to the microprocessor.
- In the middle of the T3 state RD' goes high and disables the I/O read operation.
- The data which was obtained from the I/O is then decoded.

I/O Write Cycle:

The I/O write cycle is executed by the processor to write a data byte to I/O port or to a peripheral, which is I/O mapped in the system. The processor takes three T-states to execute this machine cycle.

T1 state:

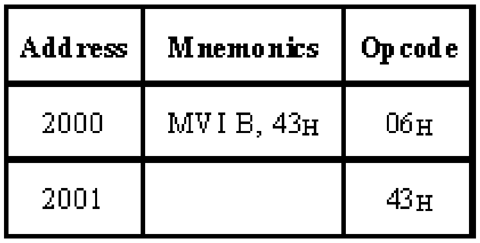
- The higher order address bus (A8-A15) and lower order address and data multiplexed (AD0-AD7) bus.
- ALE goes high so that the memory latches the (AD0-AD7) so that complete 16-bit address are available.

- The microprocessor identifies the I/O read machine cycle from the status signals IO/M'=1, S1=0, S0=1. This condition indicates the I/O read cycle.
  T2 state:
- Selected memory location is placed on the (D0-D7) of the A/D multiplexed bus. WR' goes LOW
  T3 State:
- In the middle of the T3 state WR' goes high and disables the I/O write operation. The data which was obtained from the I/O is then decoded.

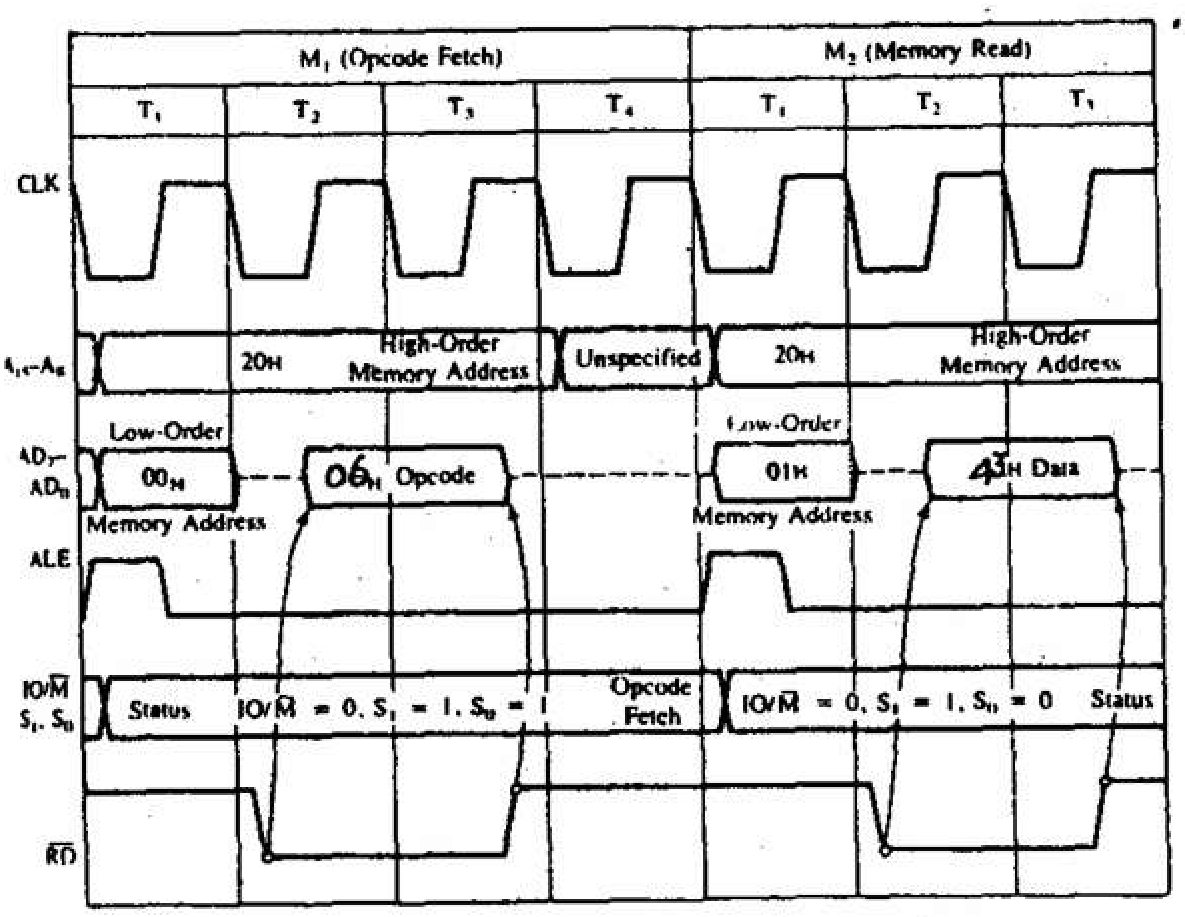
3.4 DRAW A NEAT SKETCH FOR THE TIMING DIAGRAM FOR 8085 INSTRUCTION:

Timing diagram for MVI B, 43H.

- Fetching the Opcode 06H from the memory 2000H. (OF machine cycle)
- Read (move) the data 43H from memory 2001H. (memory read)



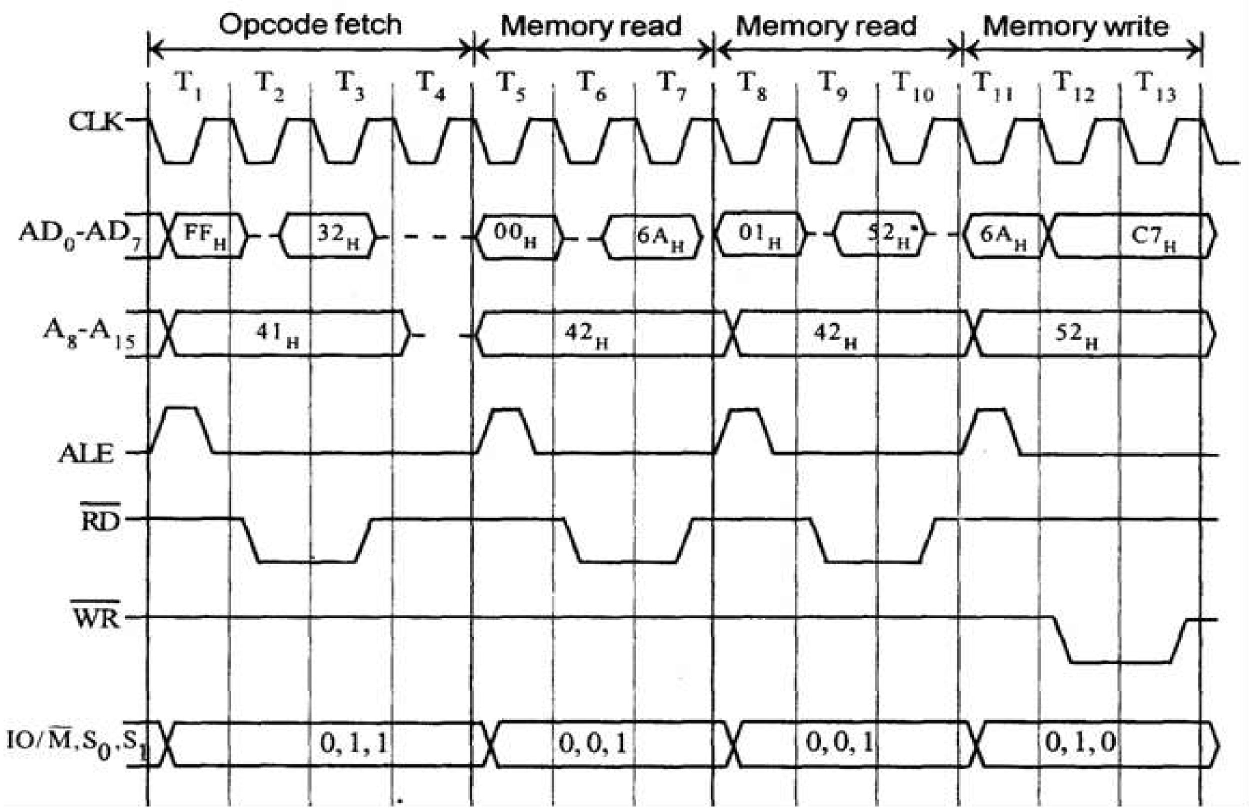
Address	Mnemonics	Opcode
2000	MVI B, 43$_H$	06$_H$
2001		43$_H$



Timing diagram for STA 526AH.

- STA means Store Accumulator -The contents of the accumulator is stored in the specified address (526A).
- The opcode of the STA instruction is said to be 32H. It is fetched from the memory 41FFH- OF machine cycle
- Then the lower order memory address is read (6A). - Memory Read Machine Cycle
- Read the higher order memory address (52).- Memory Read Machine Cycle
- The combination of both the addresses are considered and the content from accumulator is written in 526A. - Memory Write Machine Cycle
- Assume the memory address for the instruction and let the content of accumulator is C7H. So, C7H from accumulator is now stored in 526A.

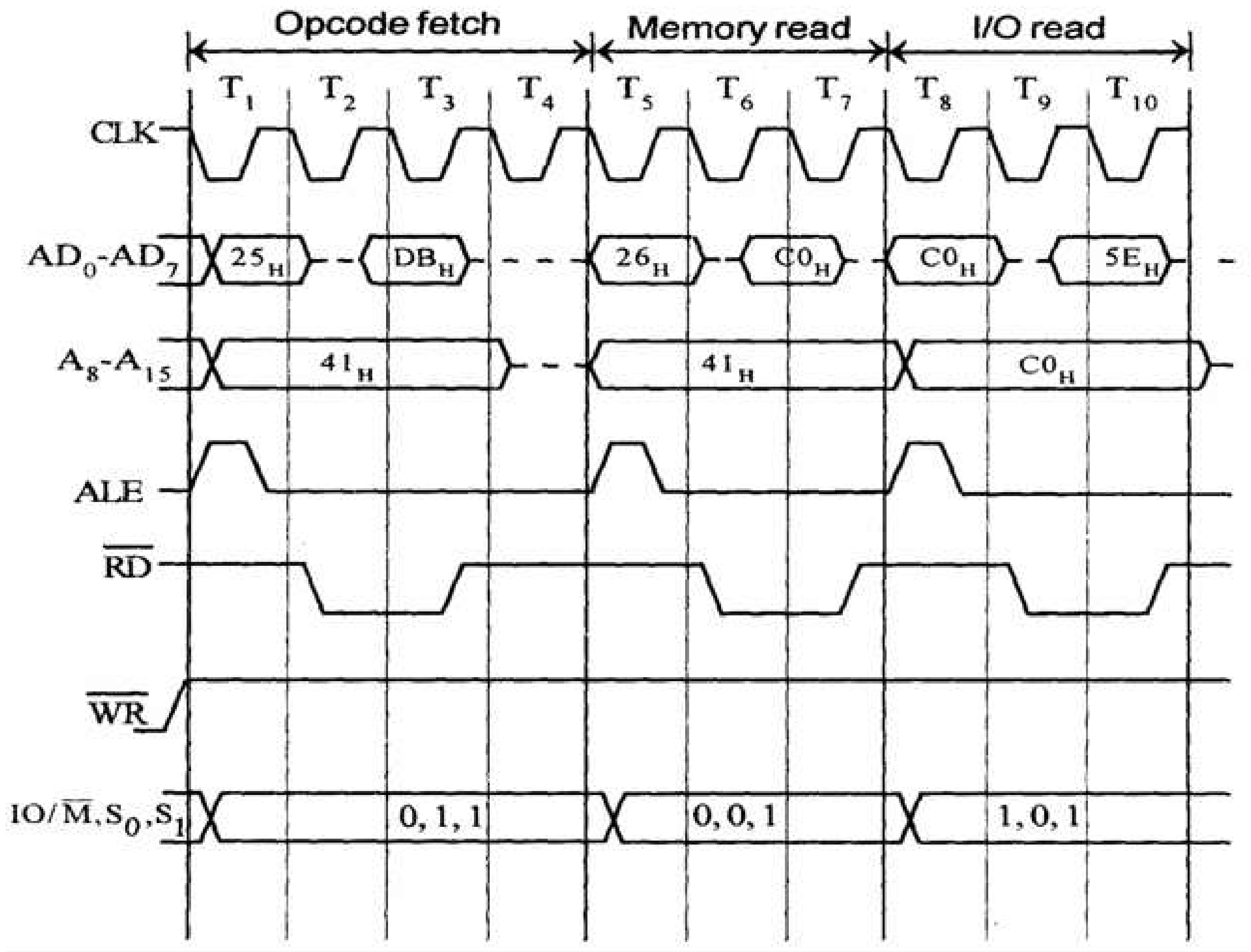
Address	Mnemonics	Opcode
41FF	STA 526A$_H$	32$_H$
4200		6A$_H$
4201		52$_H$



Timing diagram for IN C0H.

- Fetching the Opcode DBH from the memory 4125H.
- Read the port address C0H from 4126H.
- Read the content of port C0H and send it to the accumulator.
- Let the content of port is 5EH.

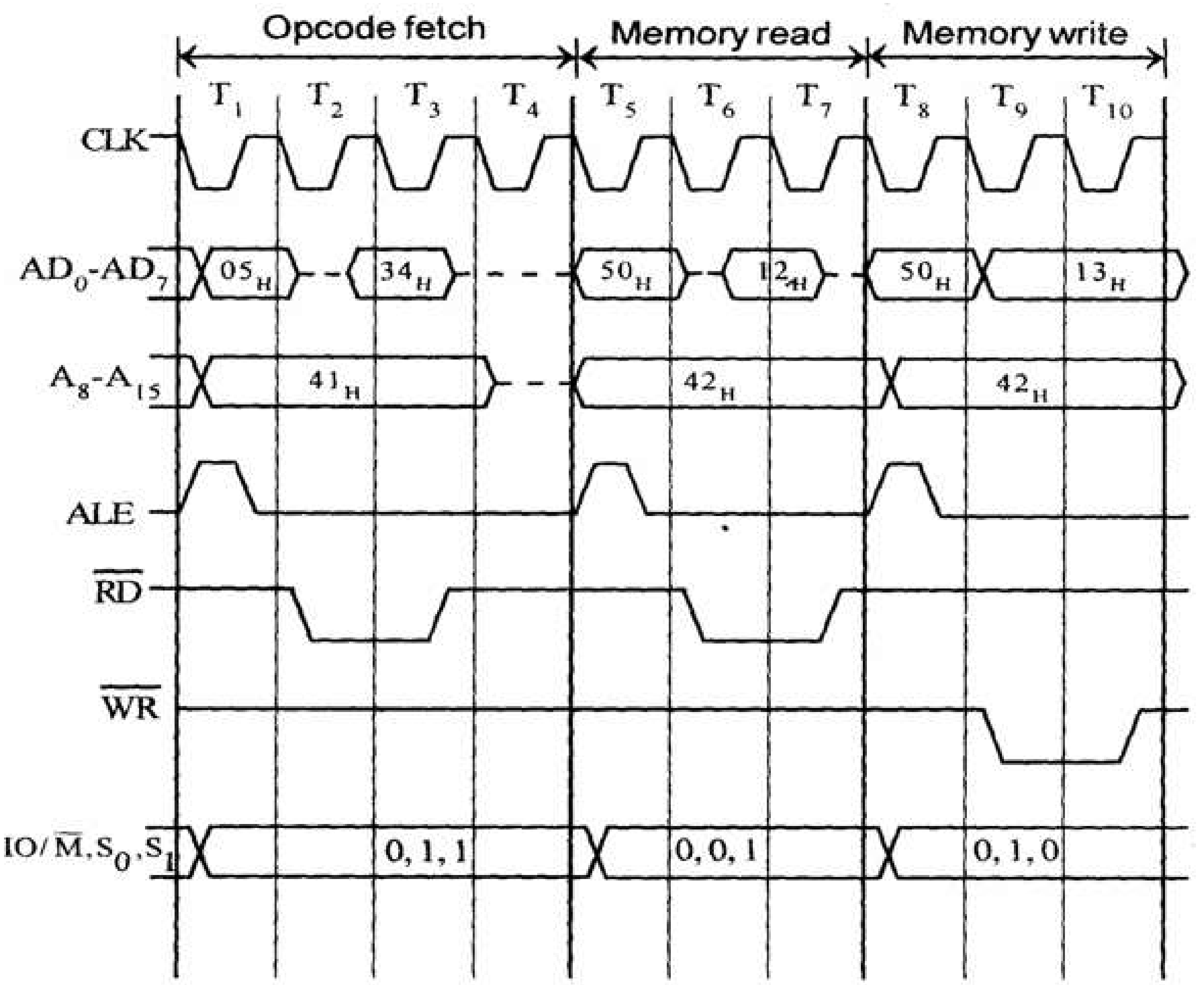
Address	Mnemonics	Opcode
4125	IN C0$_H$	DB$_H$
4126		C0$_H$



Timing diagram for INR M

- Fetching the Opcode 34H from the memory 4105H. (OF cycle)
- Let the memory address (M) be 4250H. (MR cycle -To read Memory address and data)
- Let the content of that memory is 12H.
- Increment the memory content from 12H to 13H. (MW machine cycle)

Address	Mnemonics	Opcode
4105	INR M	34_H


Opcode fetch
Memory read
Memory write
T1 T2 T3 T4 T5 T6 T7 T8 T9 T10
CLK
AD0-AD7
05H
34H
50H
12H
50H
13H
A8-A15
41H
42H
42H
ALE
RD
WR
IO/M, S0, S1
0, 1, 1
0, 0, 1
0, 1, 0

UNIT-4: MICROPROCESSOR BASED SYSTEM DEVELOPMENT AIDS

4.1 CONCEPT TO INTERFACING:

- We know that a microprocessor is the CPU of a computer. A microprocessor can perform some operation on a data and give the output. But to perform the operation we need an input to enter the data and an output to display the results of the operation. So we are using a keyboard and monitor as Input and output along with the processor. Microprocessors engineering involves a lot of other concepts and we also interface memory elements like ROM, EPROM to access the memory.
- Interfacing a microprocessor is to connect it with various peripherals to perform various operations to obtain a desired output.

INTERFACING TYPES:

There are two types of interfacing in 8085 processor.

- Memory Interfacing.
- I/O interfacing.

Purpose of interfacing:

- The interfacing process involves matching the memory requirements with the microprocessor signals.
- The interfacing circuit therefore should be designed in such a way that it matches the memory signal requirements with the signals of the microprocessor.
- For example for carrying out a READ process, the microprocessor should initiate a read signal which the memory requires to read a data.
- In simple words, the primary function of a memory interfacing circuit is to aid the microprocessor in reading and writing a data to the given register of a memory chip.

Disadvantages of interfacing:

- The main disadvantage with this interfacing is that the microprocessor can perform only one function.
- It functions as an input device if it is connected to buffer.
- It function as an output device if it is connected to latch.
- Thus the capability is very limited in this type of interfacing.

Types of Communication Interface

There are two ways in which a microprocessor can connect with outside world or other memory systems.

1. Serial Communication Interface

2. Parallel Communication interface

Serial Communication Interface:

- In serial communication interface, the interface gets a single byte of data from the microprocessor and sends it bit by bit to other system serially
- The interface also receives data bit by bit serially from the external systems and converts the data into a single byte and transfers it to the microprocessor.

Parallel Communication Interface:

- This interface gets a byte of data from microprocessor and sends it bit by bit to the other systems in simultaneous or parallel.
- The interface also receives data bit by bit simultaneously from the external system and converts the data into a single byte and transfers it to microprocessor.

4.2 MEMORY MAPPING & I/O MAPPING:

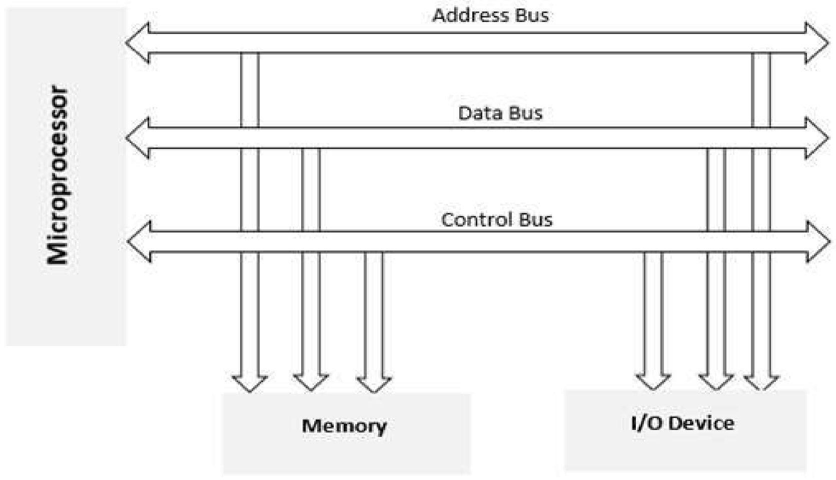
MEMORY MAPPING:

- Memory mapping is a method to expand the memory of the microprocessor.
- Being limited in memory resources, microprocessor needs to be connected to external memory devices like RAM/ROM/EEPROM.
-
- The interfacing between the microprocessor and the memory device by connecting the data and address bus is called memory mapping.

I/O INTERFACING:

- I/O Interfacing is achieved by connecting keyboard (input) and display monitors (output) with the microprocessor.
- We know that keyboard and Displays are used as communication channel with outside world. So it is necessary that we interface keyboard and displays with the microprocessor. This is called I/O interfacing. In this type of interfacing we use latches and buffers for interfacing the keyboards and displays with the microprocessor.

Block diagram of memory and I/O interfacing



I/O Mapping in 8085 Microprocessor:

I/O interfacing:

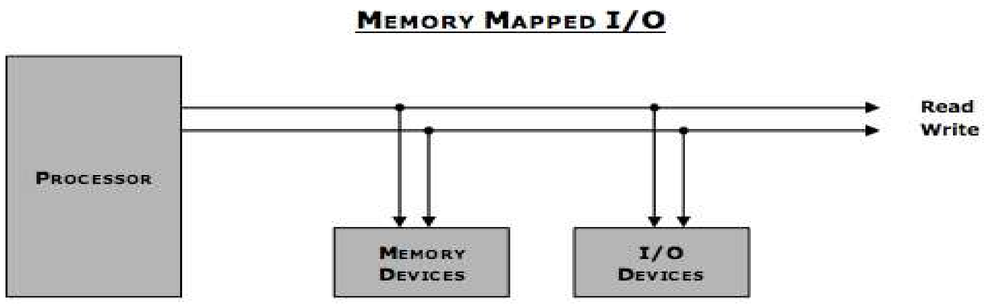
There are two methods of interfacing the Input / Output devices with the microprocessor. They are,

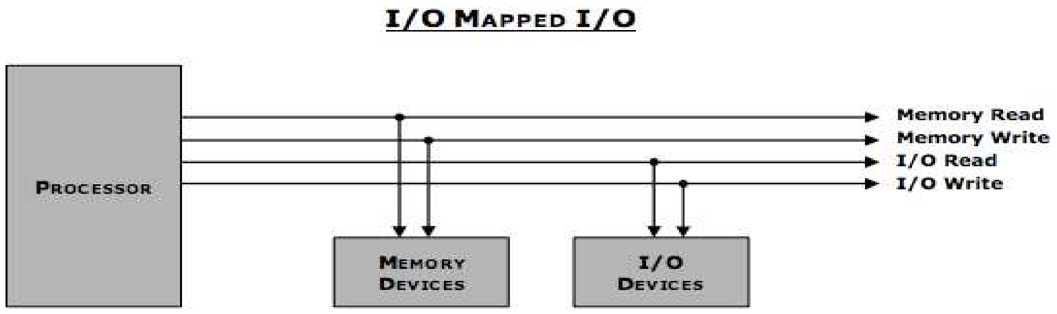
1) Memory mapped I/O and

2) I/O mapped I/O.

	MEMORY MAPPED I/O	I/O MAPPED I/O
1	**I/O devices are mapped into memory space.**	**I/O devices are mapped into I/O space.**
2	I/O devices are allotted **memory addresses**.	I/O devices are allotted **I/O addresses**.
3	Processor **does not differentiate** between memory and I/O. Treats I/O devices also like memory devices.	Processor **differentiates between I/O devices and memory**. It isolates I/O devices.
4	I/O addresses are as big as memory addresses. E.g.in 8085, **I/O addresses will be 16 bit** as memory addresses are also 16-bit.	I/O addresses are smaller than memory addresses. E.g. in 8085, **I/O addresses will be 8 bit** though memory addresses are 16-bit.

5	This allows us to increase the number of I/O devices. E.g. in 8085, we can access up to **2^{16} = 65536 I/O devices**.	This allows us to access limited number of I/O devices. E.g. in 8085, we can access only up to **2^{8} = 256 I/O devices**.
6	We can transfer data from I/O devices using **any instruction like MOV** etc.	We can transfer data from I/O device using **dedicated I/O instructions like IN and OUT ONLY**.
7	Data can be transferred using **any register** of the processor.	Data can be transferred only using a fixed register. E.g. in 8085 only **"A" register**.
8	We need **only two control** signals in the system: Read and Write.	We need **four control signals**: Memory Read, Memory Write and I/O Read and I/O Write
9	Memory addresses are big so **address decoding will be slower**.	I/O addresses are smaller so **address decoding will be faster**.
10	Address decoding will be **more complex and costly**.	Address decoding will be **simpler and cheaper**.





MEMORY MAPPED I/O:

In this method the I/O devices are treated like the memory. A part of the memory address space is used for the I/O devices. The memory mapped I/O scheme is shown in figure.

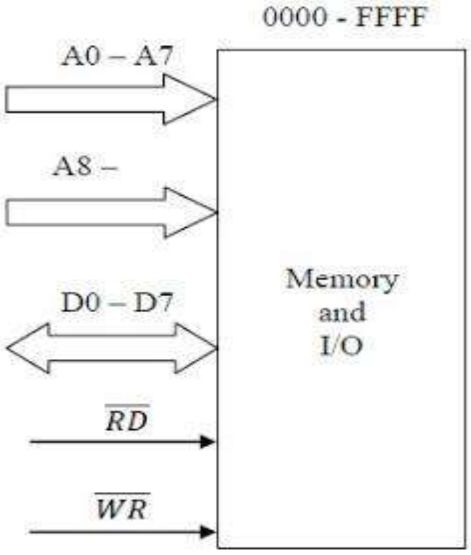


Figure: Memory mapped I/O scheme

• In memory mapped I/O scheme, the same address space is used for both memory and I/O devices.

• The microprocessor uses the sixteen address line $A_0 – A_7$ and $A_8 – A_{15}$ for the memory as well as for the I/O devices.

• The I/O devices share the address space with the memory. All the memory related instructions are used for addressing I/O devices also.

• No separate IN and OUT instructions are required in memory mapped I/O scheme.

• IO/M’ pin is not required.

Steps for memory operations (memory read and memory write):

- When the memory related instructions like LDA and STA are used, the microprocessor places the 16-bit address on the address bus.
- □□’ is activated for read operation and □□’ is activated for write operation.

Steps for I/O operations (I/O read and I/O write):

- When the memory related instructions like LDA and STA are used, the microprocessor places the 16-bit address on the address bus.
- □□’ is activated for read operation and □□’ is activated for write operation.

I/O MAPPED I/O:

In this method, I/O devices are treated as I/O devices and memory is treated as memory. Separate address space is used for memory and I/O. The I/O mapped I/O scheme is shown in figure.

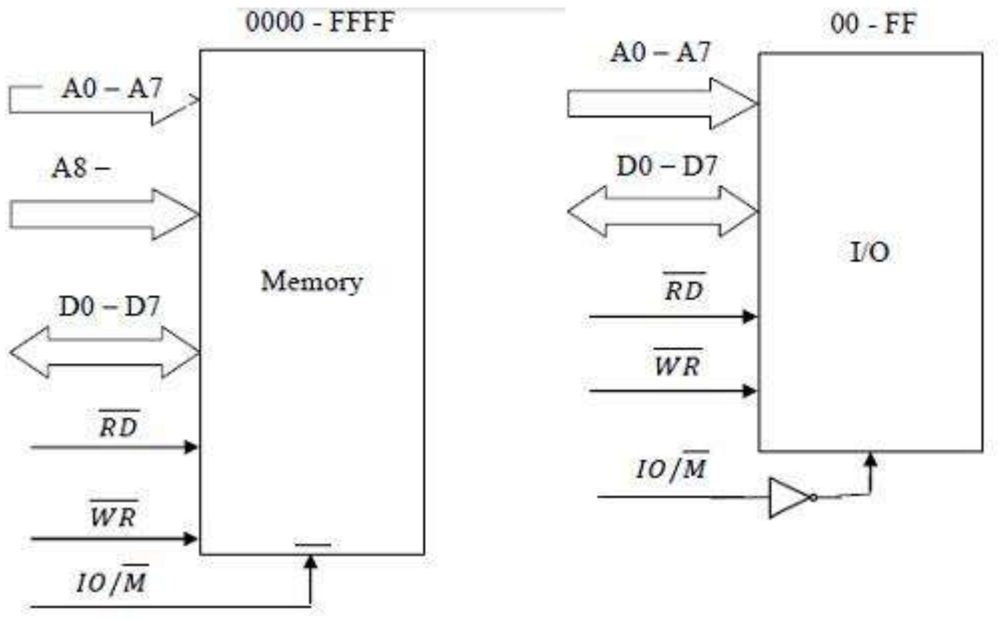


Figure: I/O mapped I/O scheme

• In I/O mapped I/O scheme, the microprocessor uses the sixteen address lines A_0 – A_7 and A_8 – A_{15} for the memory and eight address lines A_0 to A_7 to identify an input / output device.

• Here, the full address space 0000 – FFFF is used for the memory and a separate address space 00 – FF is used for the I/O devices.

• Hence, the microprocessor can address 65536 (2^{16}) memory locations 256 (2^8) input devices and 256 (2^8) output devices separately.

• IN and OUT instructions are used to activate the IO/M’ signal.

• When IO/M’ is low, the memory is selected for reading and writing operations.

• When IO/M’ is high, the I/O port is selected for reading and writing operations.

Steps for memory operations (memory read and memory write):

- When the memory related instructions like LDA and STA are used, the microprocessor places the 16-bit address on the address bus.
- The microprocessor makes the IO/M’ line low.
- The microprocessor makes the RD’ low for read operation and WR’ low for write operation.

Steps for I/O operations (I/O read and I/O write):

- 1 When the I/O related instructions like IN and OUT are used, the microprocessor places the 8-bit address on the address bus $A_0 - A_7$ as well as $A_8 - A_{15}$.
- IO/M' line is made high.
- The microprocessor makes the □□' low for read operation and □□' low for write operation.

4.3 MEMORY INTERFACING:

- While executing an instruction, there is a necessity for the microprocessor to access memory frequently for reading various instruction codes and data stored in the memory.
- The read/write operations are monitored by control signals. The microprocessor activates these signals when it wants to read from and write into memory.
- The interfacing circuit aids in accessing the memory requires some signals to read from and write to registers. Similarly the microprocessor transmits some signals for reading or writing a data.

OR

- Memory Interfacing is used when the microprocessor needs to access memory frequently for reading and writing data stored in the memory. It is used when reading/writing to a specific register of a memory chip.

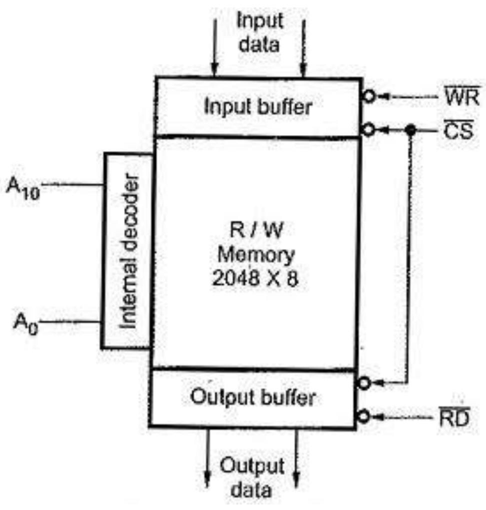
<u>The memory interfacing requires to:</u>

- ➢ Select the chip
- ➢ Identify the register
- ➢ Enable the appropriate buffer.

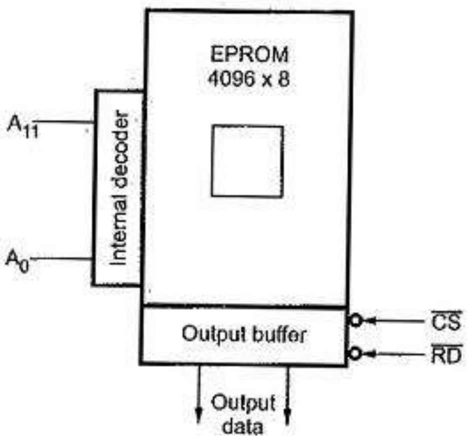
- Microprocessor system includes memory devices and I/O devices. It is important to note that microprocessor can communicate (read/write) with only one device at a time, since the data, address and control buses are common for all the devices.
- In order to communicate with memory or I/O devices, it is necessary to decode the address from the microprocessor.
- Due to this each device (memory or I/O) can be accessed independently. The following section describes common address decoding techniques.

Memory Structure and its Requirements:

As mentioned earlier, read/write memories consist of an array of registers, in which each register has unique address. The size of the memory is N x M, where N is the number of registers and M is the word length, in number of bits.



Logic diagram for RAM



Logic diagram for EPROM

INTERFACING EPROM & RAM MEMORIES:

- Microprocessor 8085 can access 64Kbytes memory since address bus is 16-bit. But it is not always necessary to use full 64Kbytes address space. The total memory size depends upon the application.
- Generally EPROM (or EPROMs) is used as a program memory and RAM (or RAMs) as a data memory. When both, EPROM and RAM are used, the total address space 64Kbytes is shared by them.
- The capacity of program memory and data memory depends on the application.
- It is not always necessary to select 1 EPROM and 1 RAM. We can have multiple EPROMs and multiple RAMs as per the requirement of application.
- We can place EPROM/RAM anywhere in full 64 Kbytes address space. But program memory (EPROM) should be located from address 0000H since reset address of 8085 microprocessor is 0000H.
- It is not always necessary to locate EPROM and RAM in consecutive memory.
- **For example**: If the mapping of EPROM is from 0000H to OFFFH, it is not must to locate RAM from 1000H. We can locate it anywhere between 1000H and FFFFH. Where to locate memory component totally depends on the application

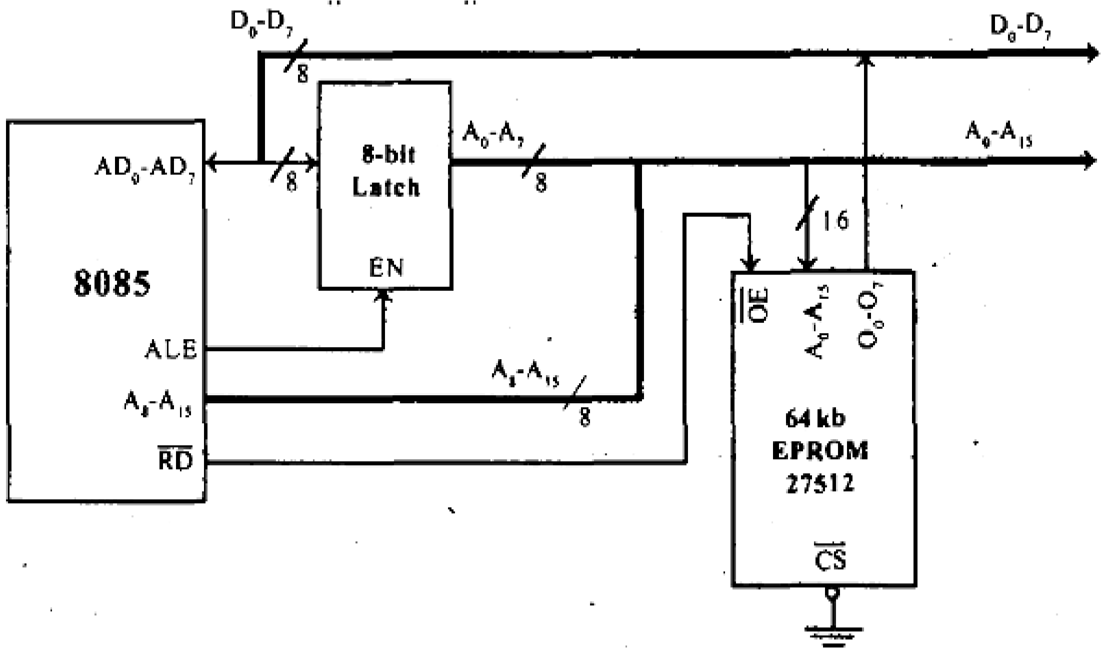
<u>EXAMPLES OF MEMORY INTERFACING</u>

EXAMPLE-1

Consider a system in which the full memory space 64kb is utilized for EPROM memory. Interface the EPROM with 8085 processor.

- The memory capacity is 64 Kbytes. i.e.
- 2^n = 64 x 1000 bytes where n = address lines.

- So, n = 16.
- In this system the entire 16 address lines of the processor are connected to address input pins of memory IC in order to address the internal locations of memory.
- The chip select (CS) pin of EPROM is permanently tied to logic low (i.e., tied to ground).
- Since the processor is connected to EPROM, the active low RD pin is connected to active low output enable pin of EPROM.
- The range of address for EPROM is 0000H to FFFFH.

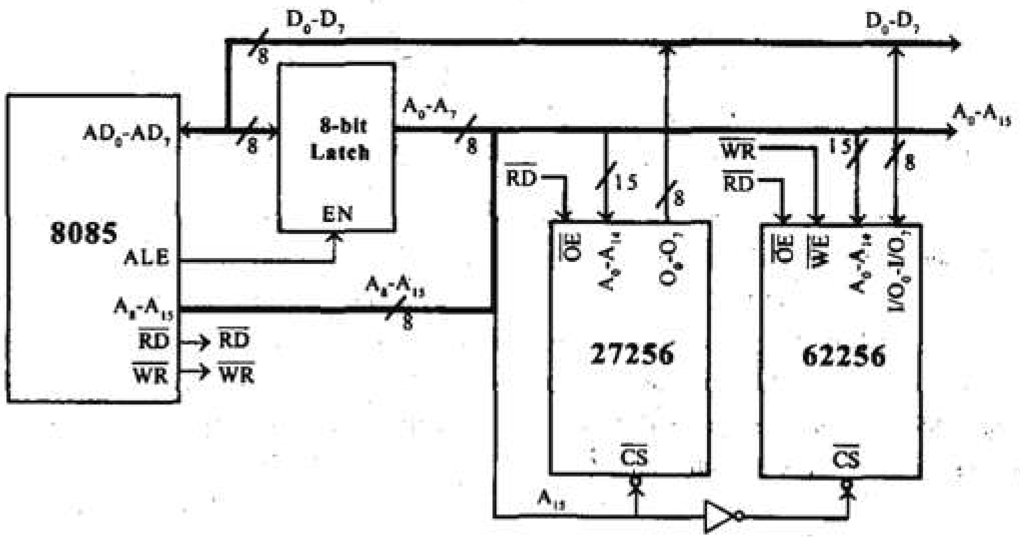


Interfacing 64Kb EPROM with 8085

EXAMPLE-2

Consider a system in which the available 64kb memory space is equally divided between EPROM and RAM. Interface the EPROM and RAM with 8085 processor.

- Implement 32kb memory capacity of EPROM using single IC 27256.
- 32kb RAM capacity is implemented using single IC 62256.
- The 32kb memory requires 15 address lines and so the address lines A0 - A14 of the processor are connected to 15 address pins of both EPROM and RAM.
- The unused address line A15 is used as to chip select. If A15 is 1, it select RAM and if A15 is 0, it select EPROM.
- Inverter is used for selecting the memory.
- The memory used is both Ram and EPROM, so the low RD and WR pins of processor are connected to low WE and OE pins of memory respectively.
- The address range of EPROM will be 0000H to 7FFFH and that of RAM will be 7FFFH to FFFFH.

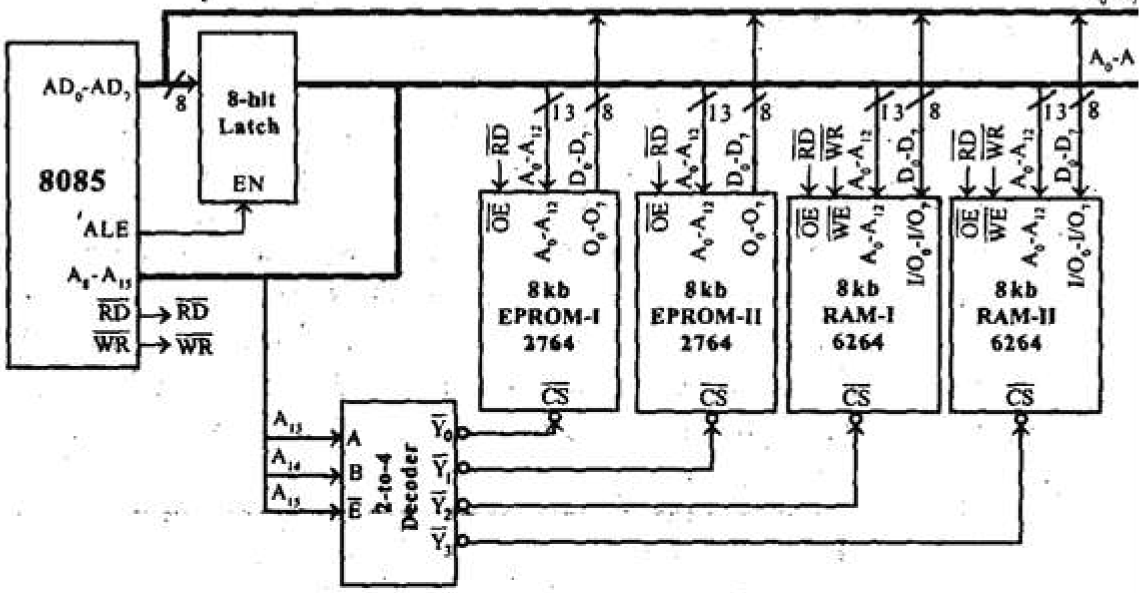


Interfacing 32Kb EPROM and 32Kb RAM with 8085

EXAMPLE-3

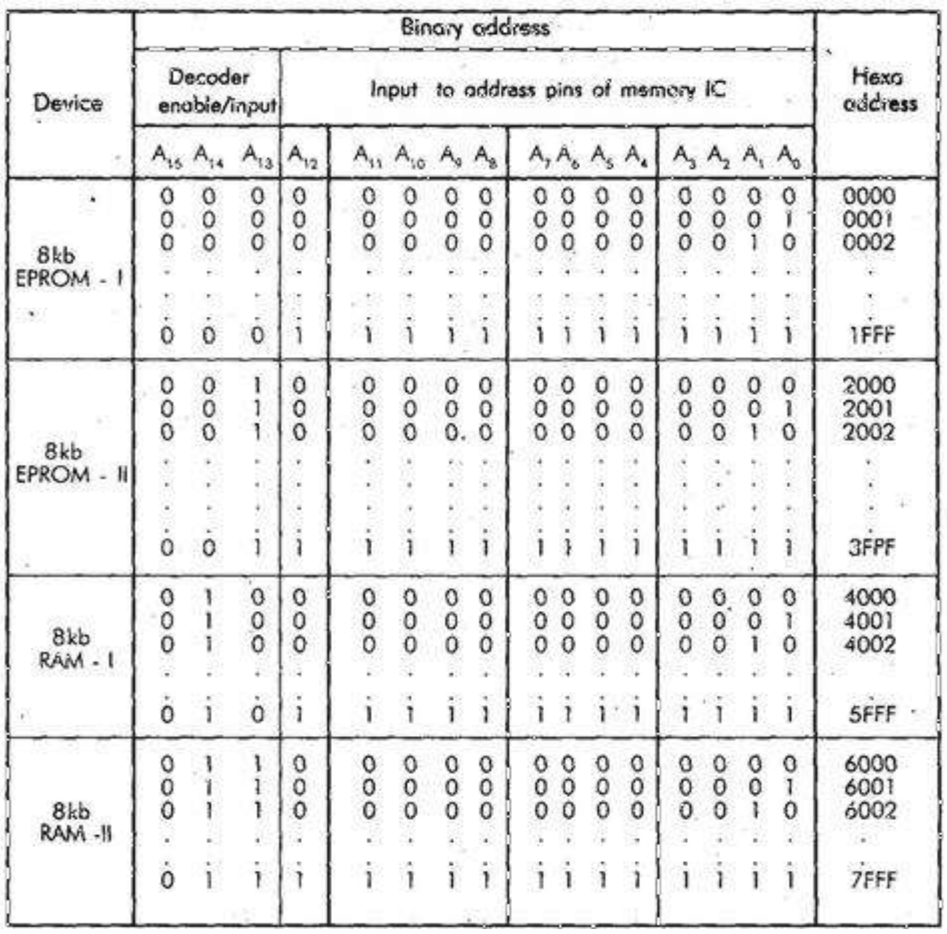
Consider a system in which 32kb memory space is implemented using four numbers of 8kb memory. Interface the EPROM and RAM with 8085 processor.

- The total memory capacity is 32Kb. So, let two number of 8kb n memory be EPROM and the remaining two numbers be RAM.
- Each 8kb memory requires 13 address lines and so the address lines A0- A12 of the processor are connected to 13 address pins of all the memory.
- The address lines and A13 - A14 can be decoded using a 2-to-4 decoder to generate four chip select signals.
- These four chip select signals can be used to select one of the four memory IC at any one time.
- The address line A15 is used as enable for decoder.
- The simplified schematic memory organization is shown.



Interfacing 16Kb EPROM and 16Kb RAM with 8085

➢ The address allotted to each memory IC is shown in following table.

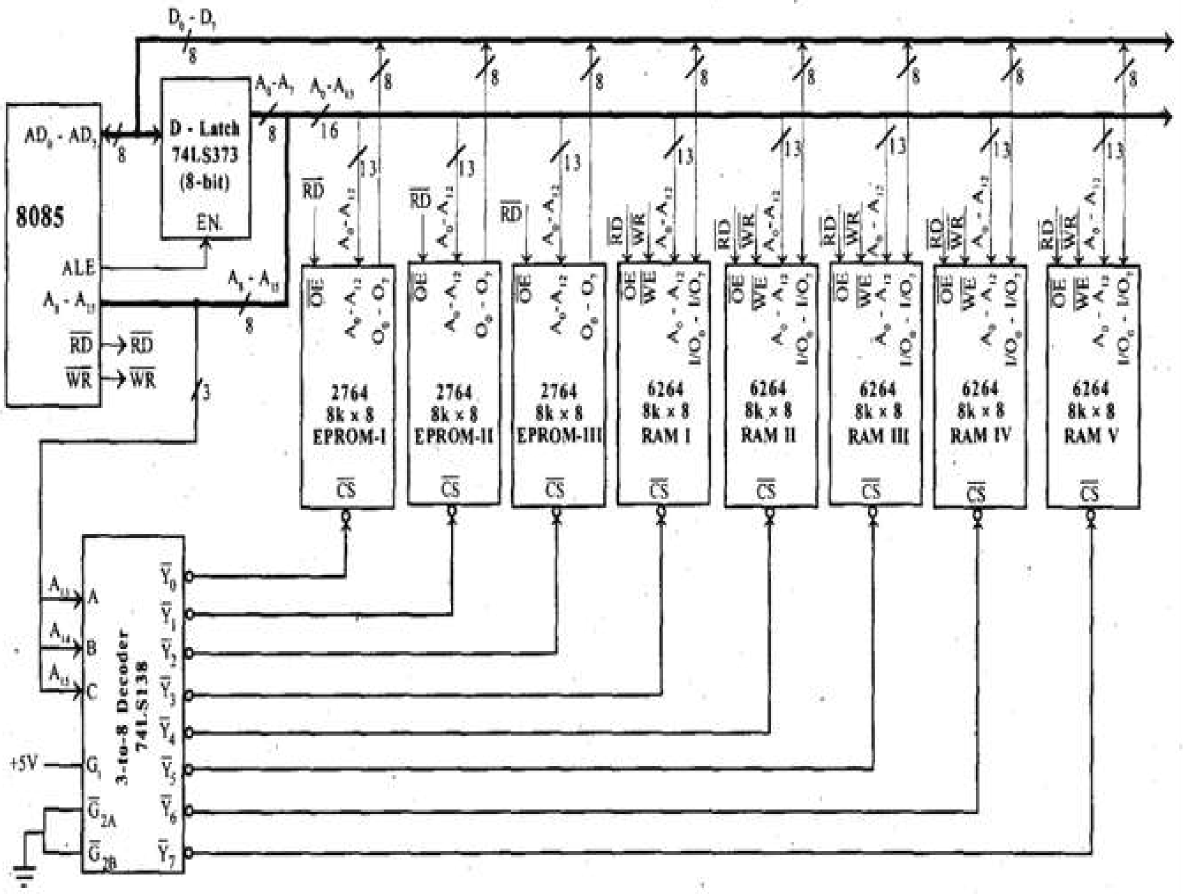


Device	Binary address																Hexa address
	Decoder enable/input			Input to address pins of memory IC													
	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
8kb EPROM - I	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0001
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0002
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFF
8kb EPROM - II	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000
	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	2001
	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	2002
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFF
8kb RAM - I	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	4001
	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	4002
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	
	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	5FFF
8kb RAM -II	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	6000
	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	6001
	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	6002
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	7FFF

EXAMPLE-4

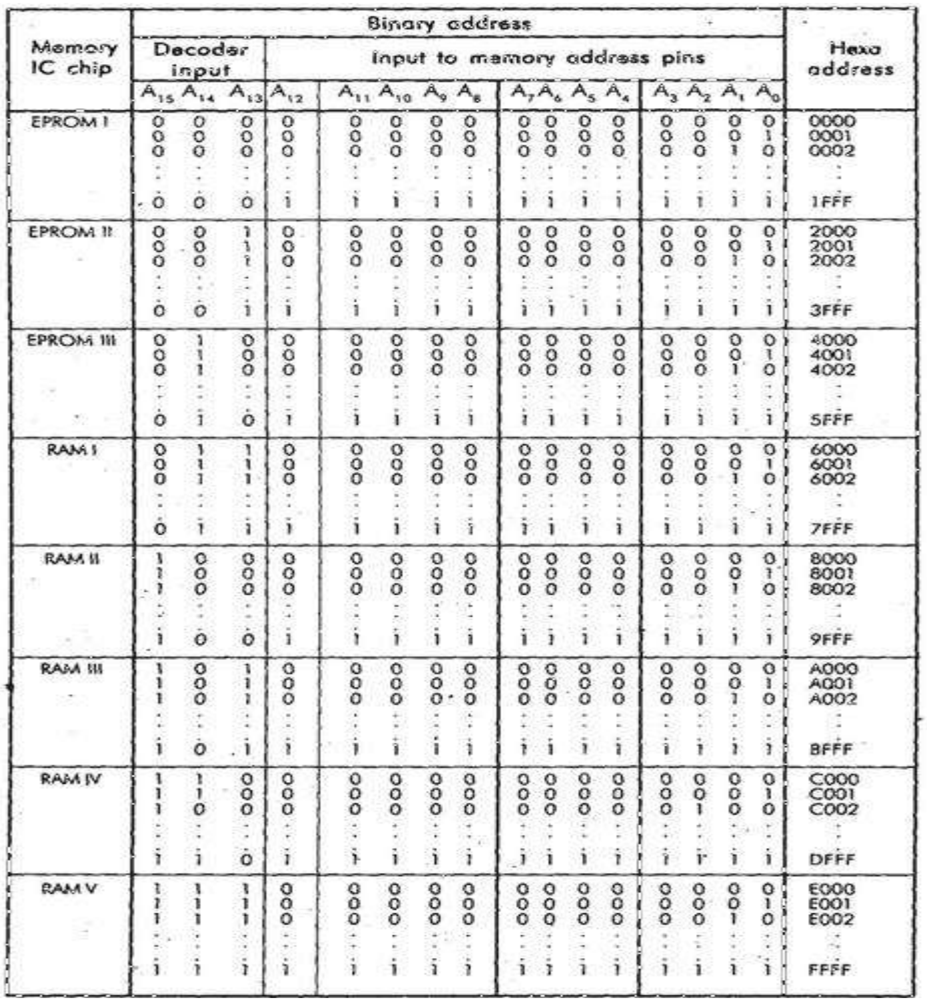
Consider a system in which the 64kb memory space is implemented using eight numbers of 8kb memory. Interface the EPROM and RAM with 8085 processor.

- The total memory capacity is 64Kb. So, let 4 numbers of 8Kb EPROM and 4 numbers of 8Kb RAM.
- Each 8kb memory requires 13 address lines. So the address line A0 - A12 of the processor are connected to 13address pins of all the memory lCs.
- The address lines A13, A14 and A15 are decoded using a 3-to-8 coder to generate eight chip select signals. These eight chip select signals can be used to select one of the eight memories at any one time.
- The memory interfacing is shown in following figure.



Interfacing 4 no. 8Kb EPROM and 4 no. 8Kb RAM with 8085

- The address allocation for Interfacing 4 no. 8Kb EPROM and 4 no. 8Kb RAM with 8085 is



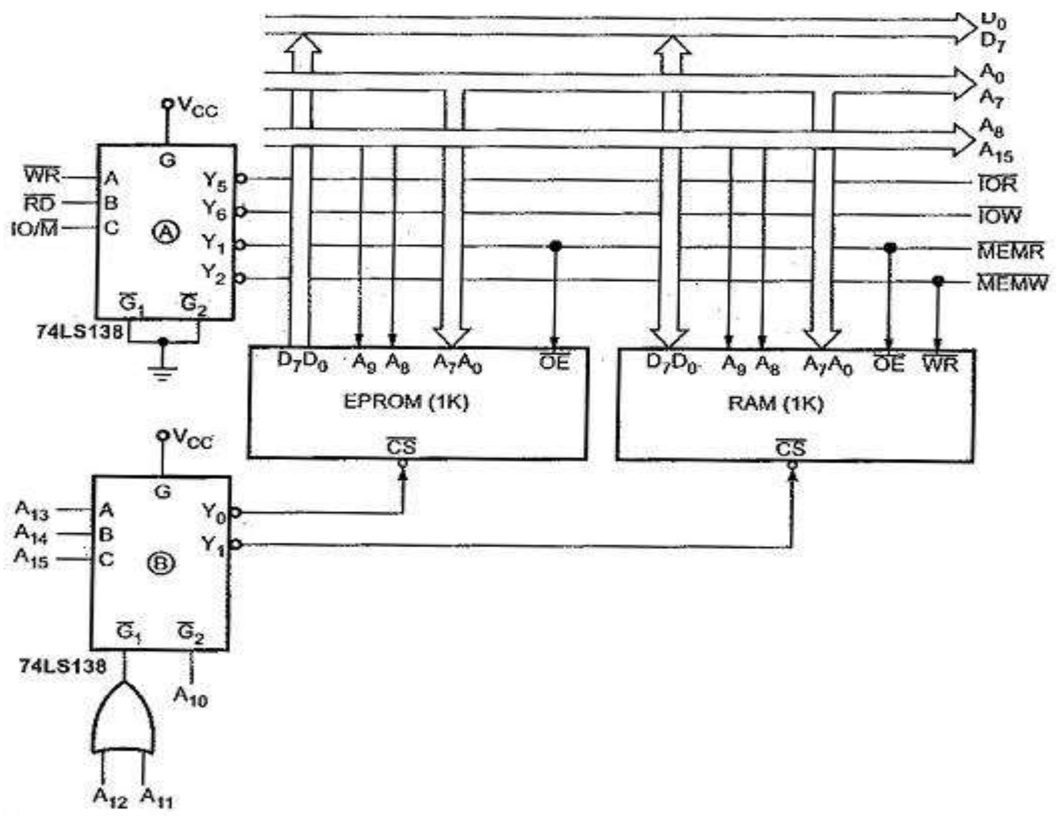
Memory IC chip	Binary address																Hexa address
	Decoder input			Input to memory address pins													
	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
EPROM I	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0001
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0002
	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:
	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFF
EPROM II	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000
	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	2001
	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	2002
	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:
	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFF
EPROM III	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	4001
	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	4002
	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:
	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	5FFF
RAM I	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	6000
	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	6001
	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	6002
	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:
	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	7FFF
RAM II	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8000
	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	8001
	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	8002
	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:
	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	9FFF
RAM III	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	A000
	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	A001
	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	A002
	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:
	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	BFFF
RAM IV	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	C000
	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	C001
	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	C002
	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:
	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	DFFF
RAM V	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	E000
	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	E001
	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	E002
	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	FFFF

4.4 ADDRESS DECODING TECHNIQUES:

- Absolute decoding/Full Decoding
- Linear decoding/Partial Decoding

Absolute decoding:

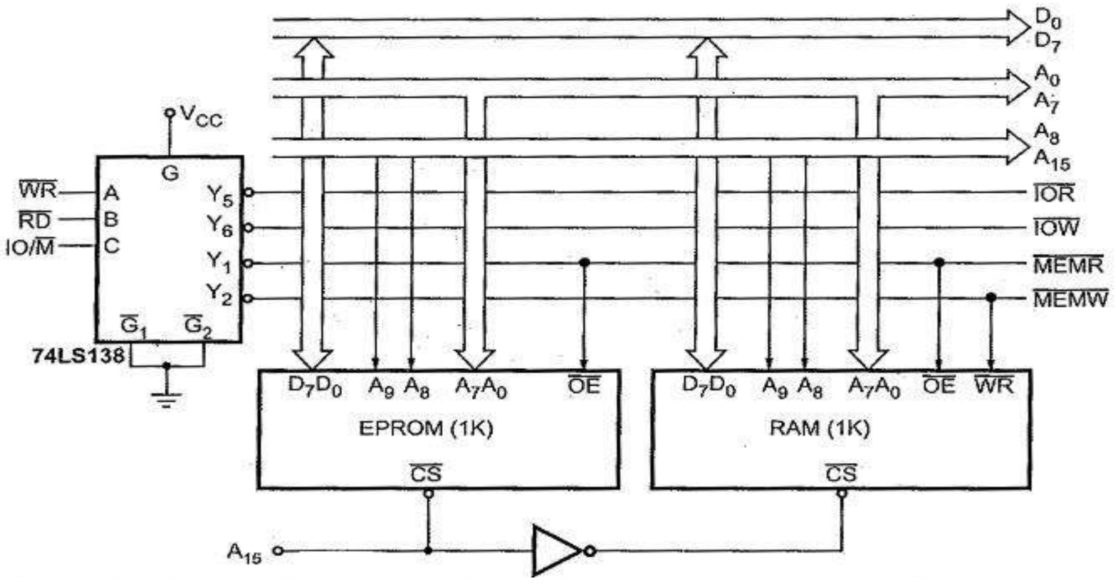
- In absolute decoding technique, all the higher address lines are decoded to select the memory chip, and the memory chip is selected only for the specified logic levels on these high-order address lines; no other logic levels can select the chip.
- This figure shows the Memory Interfacing in 8085 with absolute decoding. This addressing technique is normally used in large memory system



Absolute decoding technique diagram

Linear decoding:

- In small systems, hardware for the decoding logic can be eliminated by using individual high-order address lines to select memory chips. This is referred to as linear decoding.
- This figure shows the addressing of RAM with linear decoding technique.
- This technique is also called **partial decoding**. It reduces the cost of decoding circuit, but it has a drawback of multiple addresses (shadow addresses).



Linear decoding technique diagram

- It shows the addressing of RAM with linear decoding technique. A_{15} address line, is directly connected to the chip select signal of EPROM and after inversion it is connected to the chip select signal of the RAM.
- Therefore, when the status of A_{15} line is 'zero', EPROM gets selected and when the status of A_{15} line is 'one' RAM gets selected.
- The status of the other address lines is not considered, since those address lines are not used for generation of chip select signals.

4.5 Programmable Peripheral Interface: 8255

- 8255 is a programmable I/O device that acts as interface between peripheral devices and the microprocessor for parallel data transfer.
- 8255 PPI (programmable peripheral interface) is programmed in a way so as to have transfer of data in different conditions according to the need of the system.
- In 8255, 24 pins are assigned to the I/O ports. Basically it has three, 8-bit ports that are used for simple or interrupt I/O operations.

- The three ports are **Port A, Port B and Port C** and as each port has 8 lines, but the 8 bits of port C is divided into 2 groups of 4-bit each.
- These are given as port C lower i.e., $PC_3 - PC_0$ and port C upper i.e., $PC_7 - PC_4$.
- And are arranged in group of 12 pins each thus designated as Group A and Group B. The two modes in which 8255 can be programmed are as follows:

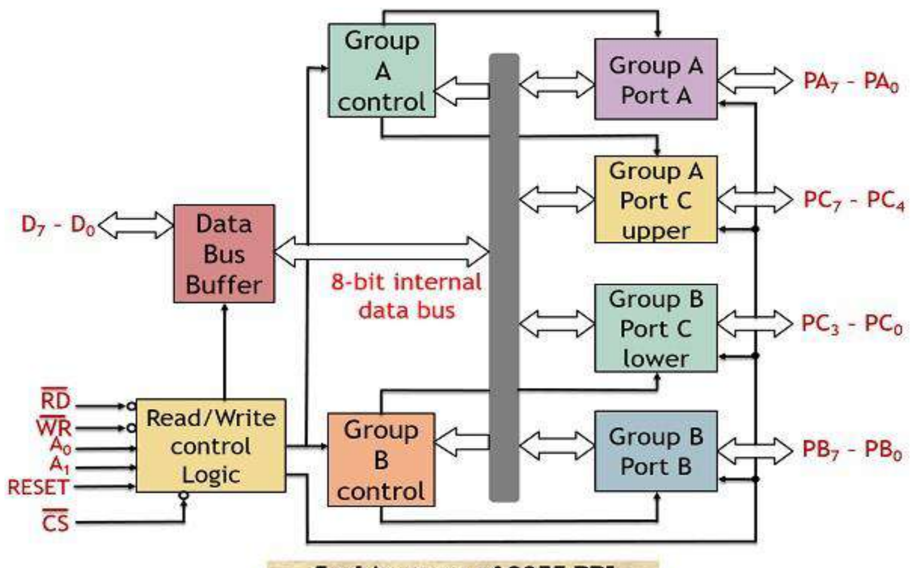
1. Bit set/reset mode
2. I/O mode

- The bits of port C gets set or reset in the BSR mode. The other mode of 8255 i.e., I/O mode is further classified into.

  Mode 0: Simple input/output
  Mode 1: Input output with handshaking
  Mode 2: Bidirectional I/O handshaking

- **Mode 1 and Mode 2** both are same but the only difference is mode 1 does not support bidirectional handshaking.
- This means if 8255 is programmed to **mode 1** input, then it will particularly be connected to an input device and performs the input handshaking with the processor.
- But if it is programmed to **mode 2** then due to bidirectional nature, the PPI will perform both input and output operation with the processor according to the command received.

Architecture of 8255 PPI:



Architecture of 8255 PPI

The figure above represents the architectural representation of 8255 PPI:

Let us understand the operation performed by each unit separately.

<u>Data bus buffer</u>:

- It is used to connect the internal bus of 8255 with the system bus so as to establish proper interfacing between the two.
- The data bus buffer allows the read/write operation to be performed from/to the CPU.
- The buffer allows the passing of data from ports or control register to CPU in case of write operation and from CPU to ports or status register in case of read operation.

<u>Read/ Write control logic</u>:

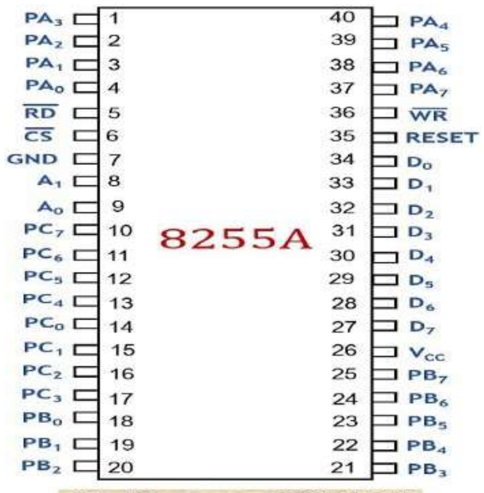
- This unit manages the internal operations of the system. This unit holds the ability to control the transfer of data and control or status words both internally and externally.
- Whenever there exists a need for data fetch then it accepts the address provided by the processor through the bus and immediately generates command to the 2 control groups for the particular operation.

<u>Group A and Group B control</u>:

- These two groups are handled by the CPU and functions according to the command generated by the CPU.
- The CPU sends control words to the group A and group B control and they in turn sends the appropriate command to their respective port.
- **Group A** the access of the **port A** and higher order bits of **port C**. While **group B** controls port B with the lower order bits of **port C.**

Pin Diagram of 8255 PPI

The figure below represents the 40 pin configuration of 8255 programmable peripheral interface:



Pin Diagram of 8255 PPI

CS:

- It stands for chip select. A low signal at this pin shows the enabling of communication between the 8255 and the processor.
- More specifically we can say that the data transfer operation gets enabled by an active low signal at this pin.

RD:

- It is the signal used for read operation.
- A low signal at this pin shows that CPU is performing read operation at the ports or status word.
- Or we can say that 8255 is providing data or information to the CPU through data buffer.

WR:

- It shows write operation. A low signal at this pin allows the CPU to perform write operation over the ports or control register of 8255 using the data bus buffer.

A_0 and A_1:

- These are basically used to select the desired port among all the ports of the 8255 and it do so by forming conjunction with RD and WR.
- It forms connection with the LSB of the address bus.

The table below shows the operation of the control signals:

A_1	A_0	RD'	WR'	CS'	Input/Output Operation
0	0	0	1	0	Port A - Data Bus
0	1	0	1	0	Port B - Data Bus
1	0	0	1	0	Port C - Data Bus
0	0	1	0	0	Data Bus - Port A
0	1	1	0	0	Data Bus - Port B

A_1	A_0	RD'	WR'	CS'	Input/Output Operation
1	0	1	0	0	Data Bus - Port C
1	1	1	0	0	Data Bus - Control register

Reset:

- It is an active high signal that shows the resetting of the PPI.
- A high signal at this pin clears the control registers and the ports are set in the input mode.
- Initializing the ports to input mode is done to prevent circuit breakdown.
- As in case of reset condition, if the ports are initialized to output mode then there exist chances of destruction of 8255 along with the processor.

PA7-PA0:

These are eight **port A** lines that acts as either latched output or buffered input lines depending upon the control word loaded into the control word register.

PC7-PC4:

- Upper nibble of port C lines. They may act as either output latches or input buffers lines. This port also can be used for generation of handshake lines in mode 1 or mode 2.

PC3-PC0:

These are the lower port C lines, other details are the same as PC7-PC4 lines.

PB0-PB7:

These are the eight port B lines which are used as latched output lines or buffered input lines in the same way as port A.

D0-D7: These are the data bus lines those carry data or control word to/from the microprocessor.

VCC:

It is a supply voltage. The 8255 requires +5V supply to operate.

GND:

It is the ground reference. If there is excessive power supply then it passed to ground.

MODES OF OPERATION:

As we have already discussed that 8255 has two modes of operation. These are as follows:

1. Bit set/reset mode
2. I/O mode

Bit Set-Reset mode:

When port C is utilized for control or status operation, then by sending an OUT instruction, each individual bit of port C can be set or reset.

I/O mode:

As we know that I/ O mode is sub-classified into 3 modes. So, let us now discuss the 3 modes here.

Mode 0: Input/output mode:

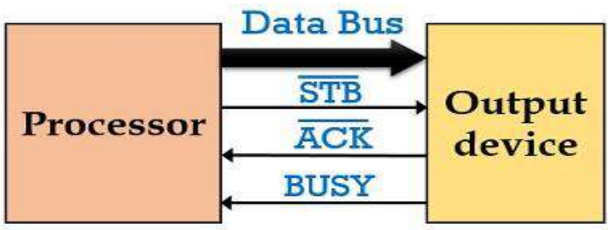
This mode is the simple input output mode of 8255 which allows the programming of each port as either input or output port. The input/output feature of mode 0 includes:

- It does not support handshaking or interrupt capability.
- The input ports are buffered while outputs are latched.

Mode 1: Input/output with handshaking:

Mode 1 of 8255 supports handshaking with the ports programmed as either input or output mode. We know that it is not necessary that all the time the data is transferred between two devices operating at same speed. So, handshaking signals are used to synchronize the data transfer between two devices that operates at different speeds.

The figure below shows the data transferring between CPU and an output device having different operating speeds:



Data Transfer using handshaking signals

- Here STB signal is used to inform the output device that data is available on the data bus by the processor.
- Here port A and port B can be separately configured as either input or output port.
- Both the port utilizes 3-3 lines of port C for handshaking signals. The rest two lines operates as input/output port.
- It supports interrupt logic.
- The data at the input or output ports are latched.

Mode 2: Bidirectional I/O port with handshaking:

- In this mode, the ports can be utilized for the bidirectional flow of information by handshaking signals.
- The pins of group A can be programmed to acts as bidirectional data bus and the port C upper (PC_7 – PC_4) are used by the handshaking signal.
- The rest 4 lower port C bits are utilized for I/O operations.
- As the data bus exhibits bidirectional nature thus when the peripheral device request for a data input only then the processor load the data in the data bus.
- Port B can be programmed in mode 0 and 1. And in mode 1 the lower bits of port C of group B are used for handshaking signals.

4.6 DAC & ADC WITH INTERFACING:

DAC INTERFACING:

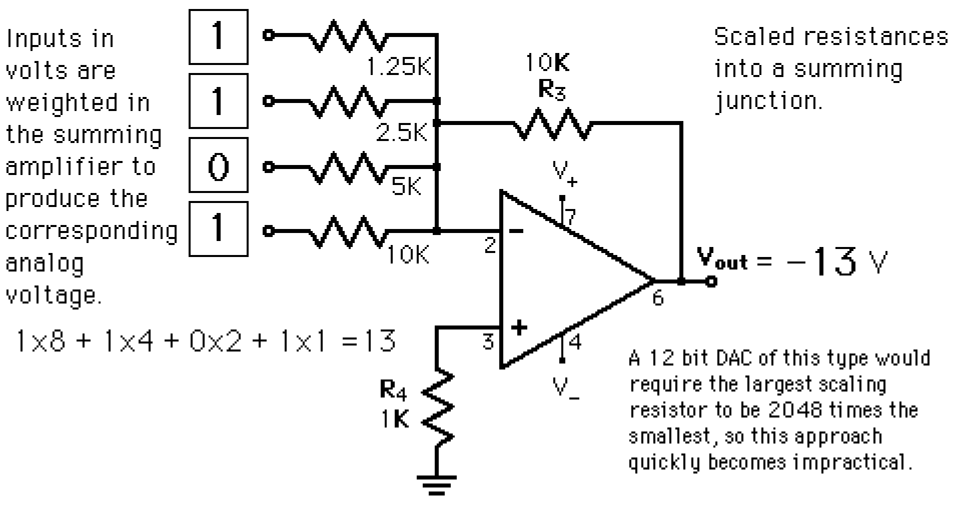
- Digital-to-Analog Conversion or simply DAC, is a device that is used to convert a digital (usually binary) code into an analog signal (current, voltage, or electric charge).
- Digital-to-analog conversion is the primary means by which digital equipment such as computer-based systems are able to translate digital data into real-world signals that are more understandable to or useable by humans, such as music, speech, pictures, video.
- It also allows digital control of machines, equipment, household appliances.
- When data is in binary form, the 0's and 1's may be of several forms such as the TTL form where the logic zero may be a value up to 0.8 volts and the 1 may be a voltage from 2 to 5 volts.
- The data can be converted to clean digital form using gates which are designed to be on or off depending on the value of the incoming signal.
- Data in clean binary digital form can be converted to an analog form by using a summing amplifier.
- Here is a simplified functional diagram of an 8-bit DAC. There are mainly two techniques used for digital to analog conversion

1. Weighted Summing Amplifier

2. R-2R Network

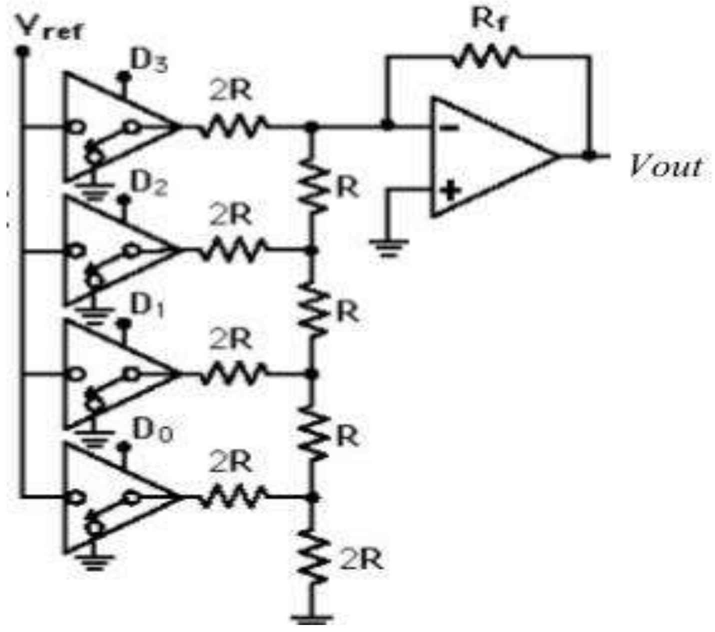
Weighted Sum DAC:

- One way to achieve D/A conversion is to use a summing amplifier.
- This approach is not satisfactory for a large number of bits because it requires too much precision in the summing resistors.
- This problem is overcome in the R-2R network DAC.

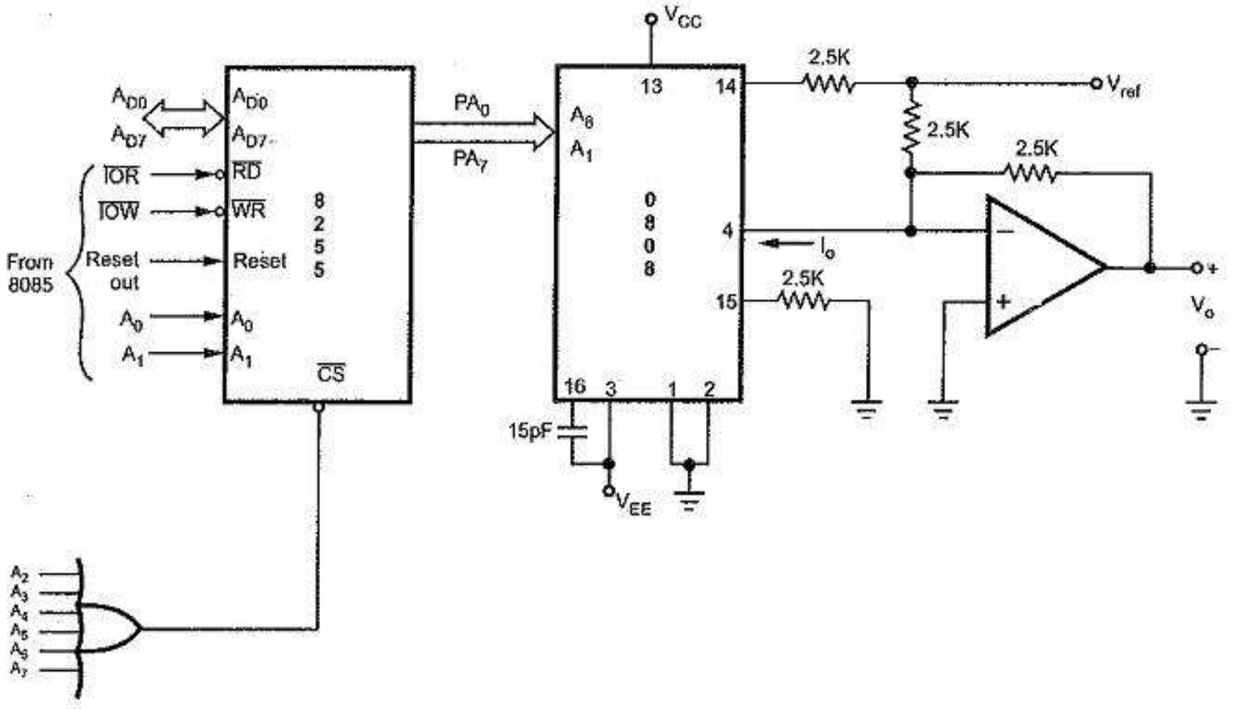

Inputs in volts are weighted in the summing amplifier to produce the corresponding analog voltage.
1
1
0
1
1.25K
2.5K
5K
10K
10K
R3
V+
7
2
3
4
6
V-
R4
1K
Vout = −13 V
1x8 + 1x4 + 0x2 + 1x1 =13
Scaled resistances into a summing junction.
A 12 bit DAC of this type would require the largest scaling resistor to be 2048 times the smallest, so this approach quickly becomes impractical.

R-2R Ladder DAC:

- The summing amplifier with the R-2R ladder of resistances shown produces the output where the D's take the value 0 or 1.
- The digital inputs could be TTL voltages which close the switches on a logical 1 and leave it grounded for a logical 0.
- This is illustrated for 4 bits, but can be extended to any number with just the resistance values R and 2R.



- The interfacing of DAC 0808 with microprocessor 8085 is shown below. Here, programmable peripheral interface, 8255 is used as parallel port to send the digital data to DAC.



Interfacing of 0808 with microprocessor

Interfacing Digital-To-Analog converter to 8085 using 8255"

- Figure below shows the interfacing of DAC 0808 with microprocessor 8085. Here, programmable peripheral interface, 8255 is used as parallel port to send the digital data to DAC.
- **I/O Map for 8255**

PORT/REGISTER	**ADDRESS**
Port A	**00**
Port B	**01**
Port C	**02**
Control Register	**03**

Program:

MVI A, 80H; Initialization -control word for 8255 to configure all ports as output Ports

OUT 03

MVI A, DATA; Load 8-bit data to be sent at the input of 0808 DAC

OUT 00; send data on port A.

A Circuit Description of DAC module:

- When chip select of DAC is enabled then DAC will convert digital input value given through portliness PB0-PB7 to analog value.
- The analog output from DAC is a current quantity. This current is converted to voltage using OPAMP based current-to-voltage converter.
- The voltage outputs (+/- 8V for bipolar 0 to 8V for unipolar mode) of OPAMP are connected to CRO to see the wave form. Port A & Port B are connected to channel 1 and channel 2 respectively.
- A reference voltage of 8V is generated using 723 and is given to Verify points of the DAC 0800. The standard output voltage will be 7.98V when FF is outputted and will be 0V when 00 is outputted.
- The Output of DAC-0800 is fed to the operational amplifier to get the final output as X OUT and Y OUT.

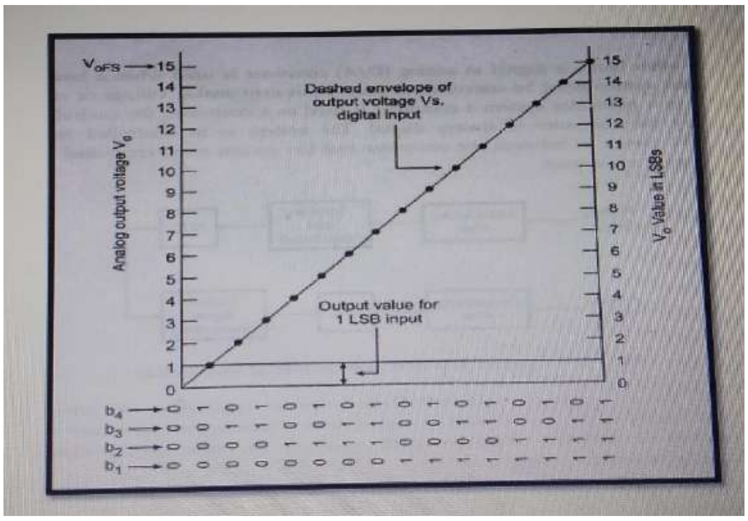


Figure shows analog output voltage v0 is plotted against all 16 possible digital input words.

Performance Parameters of DAC:

The performance parameters of DAC are:

1. **<u>Resolution</u>:**

- Resolution is defined in two ways. o Resolution is the number of different analog output values that can be provided by a DAC.
- For an n-bit DAC **Resolution = 2^n ……… (1)**

or

- Resolution is also defined as the ratio of a change in output voltage resulting from a change of 1 LSB at the digital inputs.
- For an n-bit DAC it can be given as: **Resolution= V_o F_S /2^n -1 ………(2)**
- Where, **V_o F_S** = Full scale output voltage from equation (1), we can say that, the resolution can be determined by the number of bits in the input binary word.
- For an 8-bit resolution can be given as resolution = 2^n = 2^8 = 256

- If the full scale output voltage is 10.2 V then by second definition the resolution for an 8-bit can be given as Resolution= = **V_o F_s /2^n -1 =10.2/2^8-1 =10.2/255 = 40 mV/LSB**
- Therefore, we can say that an input change of 1 LSB causes the output to change by 40 mv

2. <u>Accuracy</u>:

- lt is a comparison of actual output voltage with expected output. It is expressed in percentage. Ideally, the accuracy of DAC should be, at worst, ±1/2, of its LSB.
- If the full scale output voltage is 10.2 V then for an 8-bit DAC accuracy can be given as

  Accuracy = Vo Fs / (2^n-1)2 = 10.2/255x2 = 20 mV

ADC CONVERTER:

- It is a converter which converts analog quantity into digital quantity.
- There are many types of ADC available such as

1. RAMP type ADC
2. Dual slope ADC
3. Flash type ADC
4. Successive approximation type ADC

- Mostly the successive approximation type ADC are used.

SPECIFICATION OF ADC:

<u>Input voltage range:</u>

- The analog input can be either unipolar or bipolar.
- Unipolar means the voltage have one polarity i.e. (0 to +5V or -5V to 0).
- Bipolar means the voltage have the range from one polarity to other polarity i.e. (+5V to -5V or -10V to +10V)

<u>Output voltage range:</u>

- The resolution of ADC is defined as the smallest change input voltage can be sensed or detected at the output. Which is given by

  Resolution= <u>range of input voltage</u>
  range of output voltage

  Example→if the input voltage range from 0 to +5V and output has 8 bit then the resolution =5/2^8=19.5mv

Conversion time:

It is the time required to convert the analog input into digital output by ADC chip is known as conversion time.

Example of ADC chip:

- ADC 0800 IC
- ADC0804 IC
- ADC 0808 IC
- ADC 0816 IC

ADC INTERFACING:

The Analog to Digital Conversion is a quantizing process. Here the analog signal is represented by equivalent binary states. The A/D converters can be classified into two groups based on their conversion techniques.

- In the first technique it compares given analog signal with the initially generated equivalent signal. In this technique, it includes successive approximation, counter and flash type converters.
- In another technique it determines the changing of analog signals into time or frequency. This process includes integrator-converters and voltage-to-frequency converters.
- The first process is faster but less accurate, the second one is more accurate. As the first process uses flash type, so it is expensive and difficult to design for high accuracy.

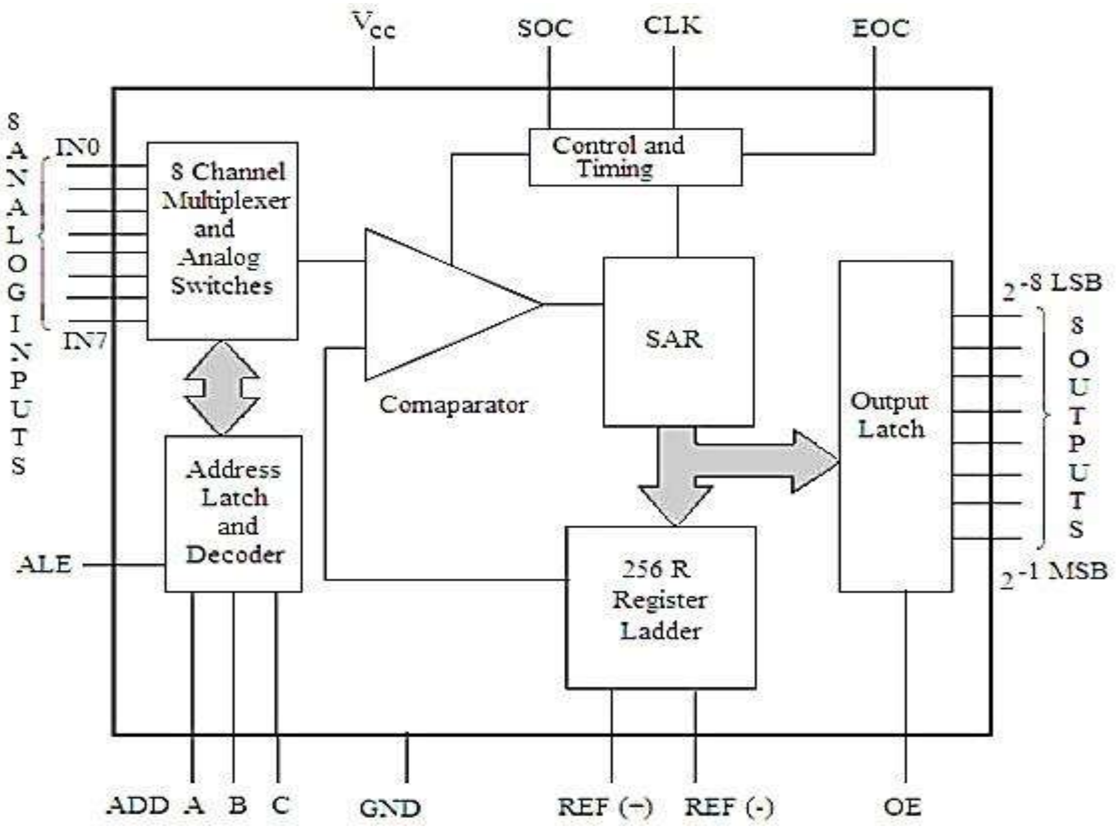
The ADC 0808/0809 Chip

- The ADC 0808/0809 is an 8-bit analog to digital converter.
- It has 8 channel multiplexer to interface with the microprocessor.
- This chip is popular and widely used ADC.
- ADC 0808/0809 is a monolithic CMOS device. This device uses successive approximation technique to convert analog signal to digital form.
- One of the main advantage of this chip is that it does not require any external zero and full scale adjustment, only +5V DC supply is sufficient.

Let us see some good features of ADC 0808/0809

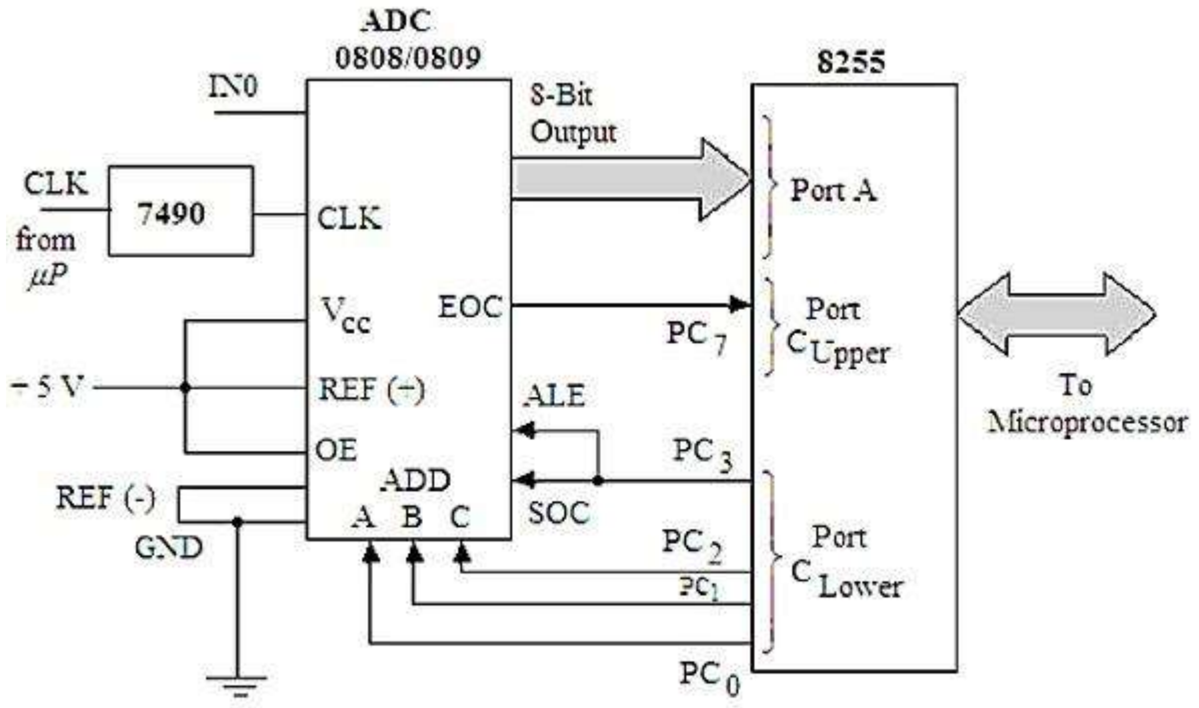
- ➢ The conversion speed is much higher
- ➢ The accuracy is also high
- ➢ It has minimal temperature dependence
- ➢ Excellent long term accuracy and repeatability
- Less power consumption

The functional block diagram of this chip is like this



Interfacing ADC with 8085 Microprocessor:

To interface the ADC with 8085, we need 8255 Programmable Peripheral Interface chip with it. Let us see the circuit diagram of connecting 8085, 8255 and the ADC converter.



- The Port A of 8255 chip is used as the input port. The PC_7 pin of Port C_{upper} is connected to the End of Conversion (EOC) Pin of the analog to digital converter. This port is also used as input port.

- The C_{lower} port is used as output port. The PC_{2-0} lines are connected to three address pins of this chip to select input channels.
- The PC_3 pin is connected to the Start of Conversion (SOC) pin and ALE pin of ADC 0808/0809.

  Now let us see a program to generate digital signal from analog data. We are using IN0 as input pin, so the pin selection value will be 00H.

MVI A, 98H; Set Port A and Cupper as input, C_{Lower} as output

OUT 03H; Write control word 8255-I to control Word register

XRA A; Clear the accumulator

OUT 02H; send the content of accumulator to Port C_{lower} to select

IN0

MVI A, 08H; Load the accumulator with 08H

OUT 02H; ALE and SOC will be 0

XRA A; Clear the accumulator

OUT 02H; ALE and SOC will be low.

READ: IN 02H; Read from EOC (PC7)

RAL: Rotate left to check C7 is 1.

JNC READ; if C7 is not 1, go to READ

IN 00H: Read digital output of ADC

STA 8000H: Save result at 8000H

HLT: Stop the program

4.7 INTERFACING SEVEN SEGMENT DISPLAYS

SEVEN SEGMENT DISPLAYS:

- A seven-segment display is a form of electronic display device for displaying decimal numerals that is an alternative to the more complex dot matrix displays.
- Seven-segment displays are widely used in digital clocks, electronic meters, basic calculators, and other electronic devices that display numerical information.
- The binary information can be displayed in the form of decimal using this seven segment display. Its wide range of applications is in microwave ovens, calculators, washing machines, radios, digital clocks etc.

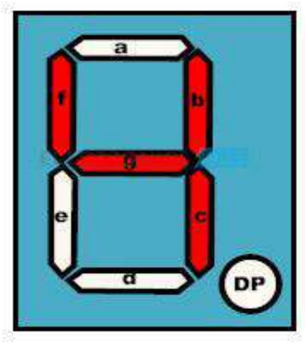
- The seven segment displays are made up of either LEDs (Light emitting diode) or LCDs (Liquid crystal display). LED or light emitting diode is P-N junction diode which emits the energy in the form of light, differing from normal P-N junction diode which emits in the form of heat.

- Liquid crystal displays (LCD) use the properties of liquid crystal for displaying. LCD will not emit the light directly. These LED's or LCD are used to display the required numeral or alphabet. Single seven segment or number of segments arranged in an order meets our requirements.

Working of Seven Segment Display:

- Seven segment display works, by glowing the required respective LEDS in the numeral. The display is controlled using pins that are left freely. Forward biasing of these pins in a sequence will display the particular numeral or alphabet. Depending on the type of seven segment the segment pins are applied with logic high or logic zero and in the similar way to the common pins also.

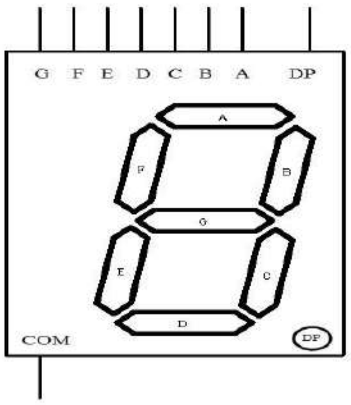
- For example to display numeral '1' segments b and c are to be switched on and the remaining segments are required to be switched off. In order to display two digits two seven segments are used.



- Depending on either the common pin is anode or cathode, seven segments are divided into following types.

INTERFACING SEVEN SEGMENT DISPLAY:

- Seven Segment Display Interfacing are generally used as numerical indicators and consists of a number of LEDs arranged in seven segments as shown in the Figure

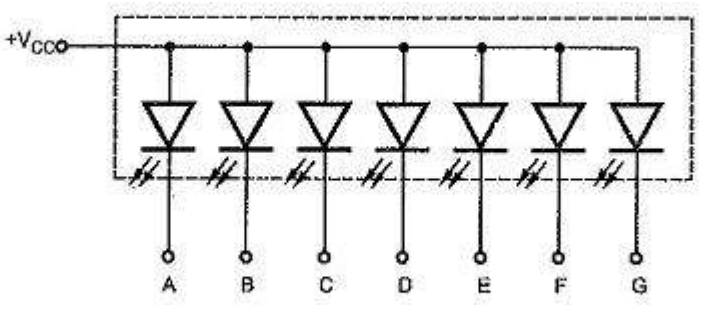


- Any number between 0 and 9 can be indicated by lighting the appropriate segments. The 7-segment displays are of two types:

  1. Common anode type
  2. Common cathode type.

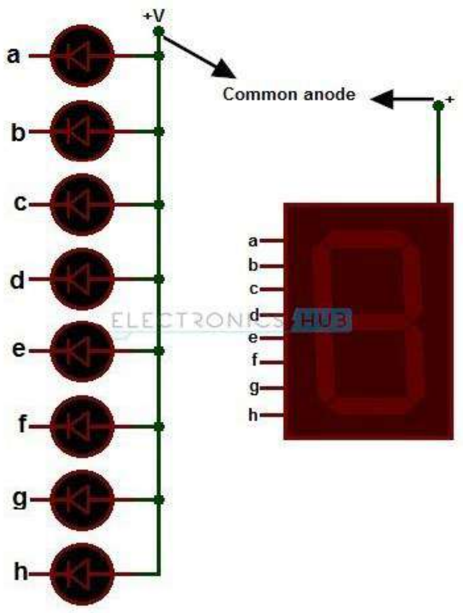
  <u>Common Anode type:</u>

- In common anode, all anodes of LEDs are connected together as shown in Fig.

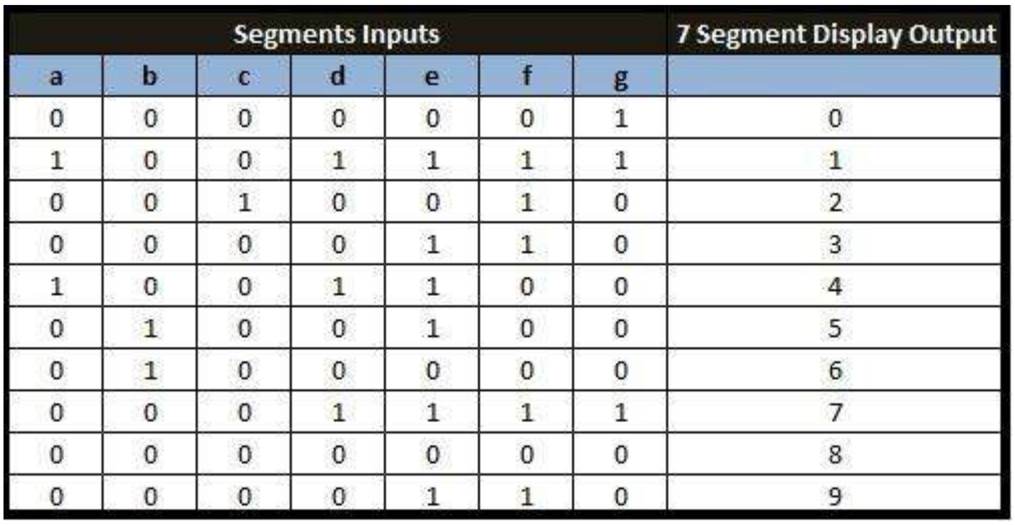


Common anode type

or



- In common anode type, all the anodes of 8 LED's are connected to the common terminal and cathodes are left free. Thus, in order to glow the LED, these cathodes have to be connected to the logic '0' and anode to the logic '1'.
- Below truth table gives the information required for driving the common anode seven segments.

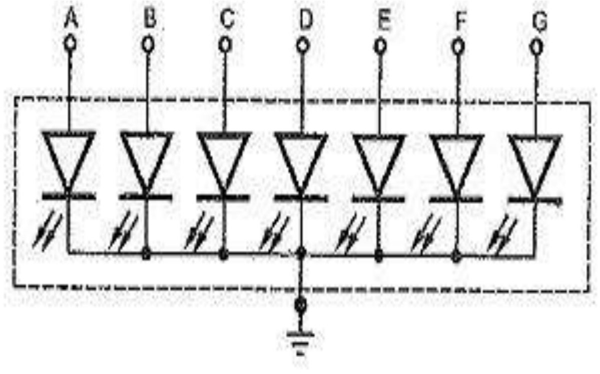


Segments Inputs							7 Segment Display Output
a	b	c	d	e	f	g	
0	0	0	0	0	0	1	0
1	0	0	1	1	1	1	1
0	0	1	0	0	1	0	2
0	0	0	0	1	1	0	3
1	0	0	1	1	0	0	4
0	1	0	0	1	0	0	5
0	1	0	0	0	0	0	6
0	0	0	1	1	1	1	7
0	0	0	0	0	0	0	8
0	0	0	0	1	1	0	9

- In order to display zero on this segment one should enable logic high on a, b, c, d, e and f segments and logic low on segment 'g'. Thus, the above table provides data on seven segments for displaying numerals from 0-9.

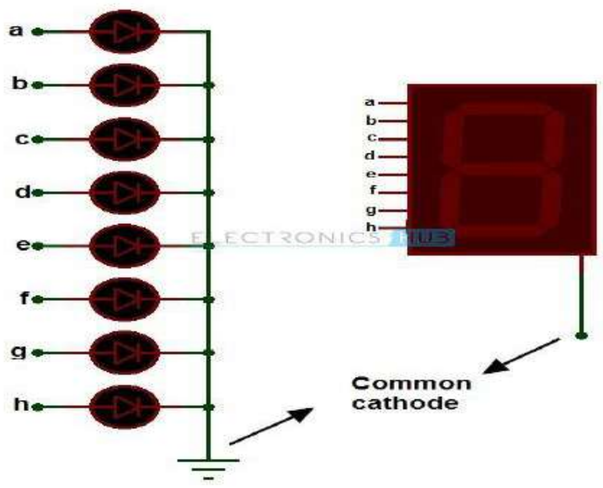
Common cathode type:

- As the name indicates cathode is the common pin for this type of seven segments and remaining 8 pins are left free. Here, logic low is applied to the common pin and logic high to the remaining pins.
- in common cathode, all cathodes are connected together, as shown in Fig

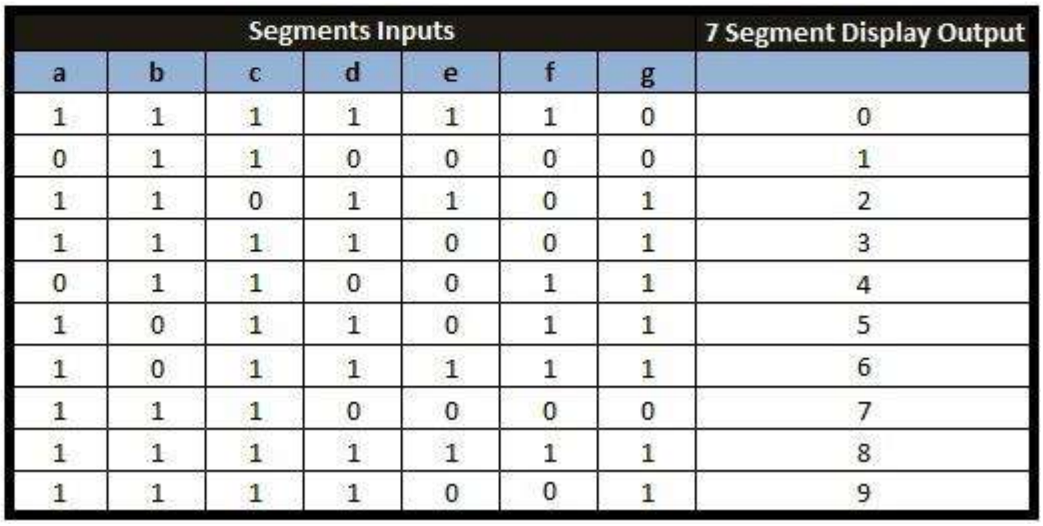


Common cathode type

Or



The truth table of seven segment display is shown below.

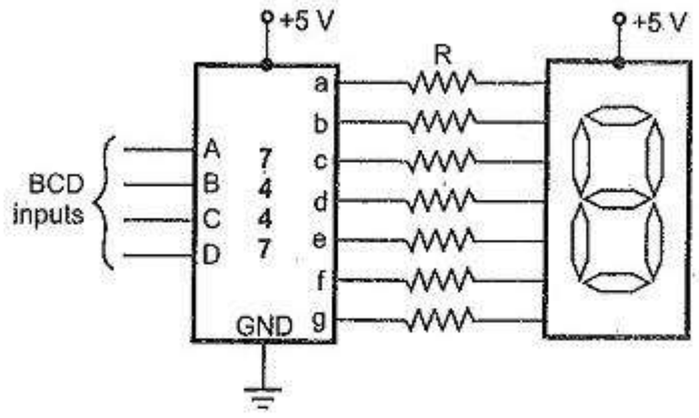


Segments Inputs							7 Segment Display Output
a	b	c	d	e	f	g	
1	1	1	1	1	1	0	0
0	1	1	0	0	0	0	1
1	1	0	1	1	0	1	2
1	1	1	1	0	0	1	3
0	1	1	0	0	1	1	4
1	0	1	1	0	1	1	5
1	0	1	1	1	1	1	6
1	1	1	0	0	0	0	7
1	1	1	1	1	1	1	8
1	1	1	1	0	0	1	9

- Above truth table shows the data to be applied to the seven segments to display the digits. In order to display digit'0' on seven segment , segments a , b , c , d , e and f are applied with logic high and segment g is applied with logic low.

Driving a Seven Segment Display:

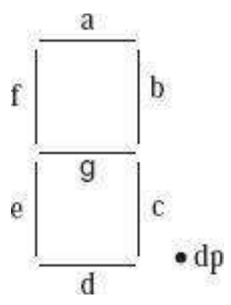
- It shows a circuit to drive a single, Seven Segment Display Interfacing, common anode LED display.
- For common anode, when anode is connected to positive supply, a low voltage is applied to a cathode to turn it on.
- Here, BCD to seven segment decoder, IC 7447 is used to apply low voltages at cathodes according to BCD input applied to 7447.
- To limit the current through LED segments resistors are connected in series with the segments.
- This circuit connection is referred to as a static display because current is being passed through the display at all times.

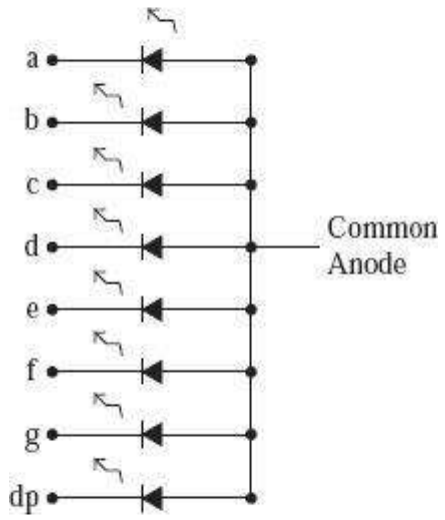


Circuit for driving single seven segment LED display

INTERFACING:

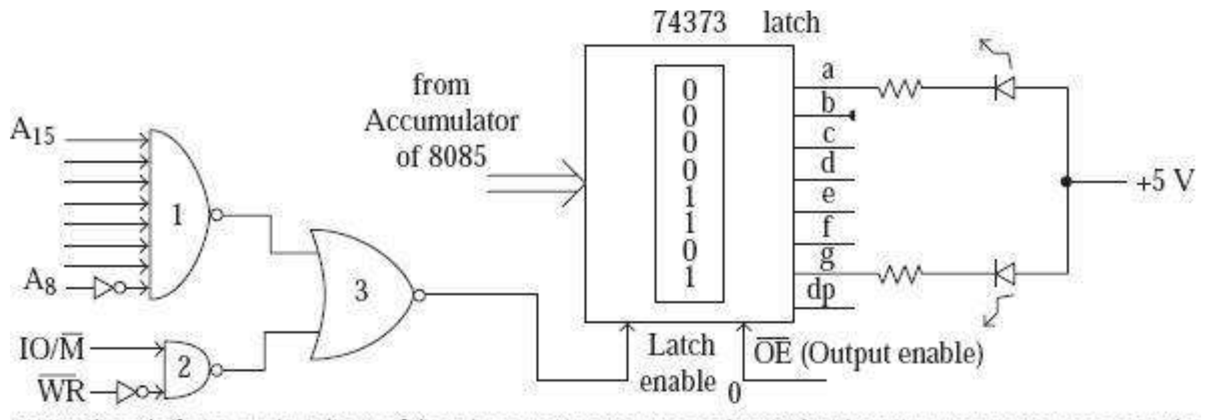
- An output device which is very common is, especially in the kit of 8085 microprocessor and it is the Light Emitting Diode consisting of seven segments.
- Moreover, we have eight segments in a LED display consisting of 7 segments which, consist of character 8 and having a decimal point just next to it.
- We denote the segments as 'a, b, c, d, e, f, g, and dp' where dp signifies '.' which is the decimal point.
- Moreover, these are LEDs or together a series of Light Emitting Diodes. We have shown the internal circuit comprising of a display of seven segment





- We have discussed the common anode-type which is 7 segmented Light Emitting Diode.
- In the LED which is common anode and is 7-segmented, here we connect all the eight LED anodes together and the eight external pin is brought to display.
- And this pin gets connected to a DC supply of +5 Volt.
- The cathode ends of the eight segments are brought out on the pins of the display.

The use of 74373 latch for interfacing a 7-segment display is shown in the following Fig.



Note: To keep the figure simple, only two of the eight segment connections are shown. The other six segment connections are similar.

- In the 74373 latch is used as an I/O mapped I/O port with the port address as FEH.
- This could be easily verified from the chip select circuit used in the figure.
- The following instructions are to be executed to display character '3' on the 7-segment display.
- The corresponding program to send 0DH to the port FEH will be -

MVI A, 0DH

OUT FEH

- Using MVI instruction we are initializing Accumulator (A) with Byte 0DH i.e. 0000 1101.
- Then it will be sent to the port FEH by the instruction OUT.

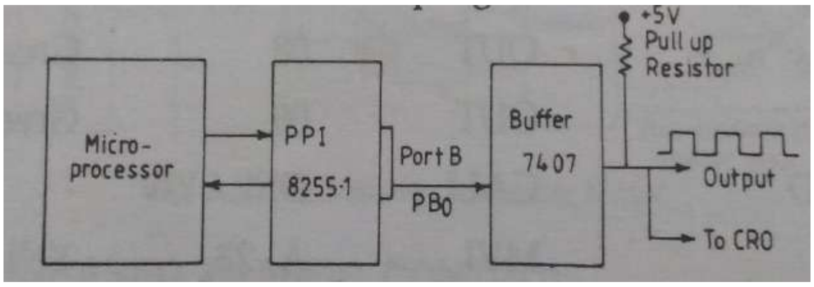
4.8 GENERATE SQUARE WAVES ON ALL LINES OF 8255:

- A square wave or pulse can easily be generated by microprocessor.
- The microprocessors sends high and then low signals to generate square wave or pulse.

- A pulse or square wave can be generated using I/O port or SOD line or timer/counter (Intel 8253).

To generate square wave or pulse using I/O port:

- To generate square wave connections are made as shown in figure below.



To generate square wave using microprocessor

- The pin PB_0 of the port B is used for taking output. This is connected to a buffer 7407. The final output is taken from the buffer terminal.
- The 8255 has been designed as general purpose programmable I/O devices, compatible with Intel microprocessor.
- It contain three 8-bit port which can be configured by software means to prove any one of 3 programmable data transfer modes available with 8255.
- The control word used in the program is 98H to make port B an output port.

PROGRAM:

MEMORY ADDRESS	**MACHINE CODES**	**LABLES**	**MNEMONICS**	**OPERANDS**	**COMMENTS**
2400	3E, 98		MVI	A,98 H	Get control word.
2402	D3, 0B		OUT	08	Initialize ports.
2404	3E, 00	LOOP	MVI	A,00	Move '00' into accumulator
2406	D3,09		OUT	09	Make PB_0 LOW
2408	CD, 00, 25		CALL	DELAY 1	Call the DELAY 1 subroutine.
240B	3E, 01		MVI	A,01	Move '01' into accumulator
240D	D3, 09		OUT	09	Make PB_0 HIGH
240F	CD, 09, 25		CALL	DELAY 2	Call the DELAY 2 subroutine.
2412	C3, 04, 24		JMP	LOOP	Jump to LOOP.
SUBROUTINES DELAY 1					
2500	06, 02		MVI	B,02	Get count for delay.
2502	05	GO	DCR	B	Decrement register B by 1.

2503	C3, 02, 25		JNZ	GO	Is B=0? No, then jump to GO.
2506	C9		RET		
DELAY 2					
2509	0E, 02		MVI	C,02	Get count for delay
250B	0D	BACK	DCR	C	Decrement register C by 1.
250C	C2, 0B, 25		JNZ	BACK	Is C=0? No, then go to BACK.
250F	C9		RET		

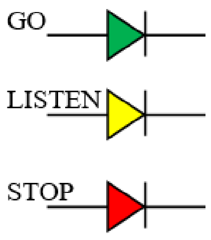
4.9 DESIGN INTERFACE A TRAFFIC LIGHT CONTROL SYSTEM USING 8255:

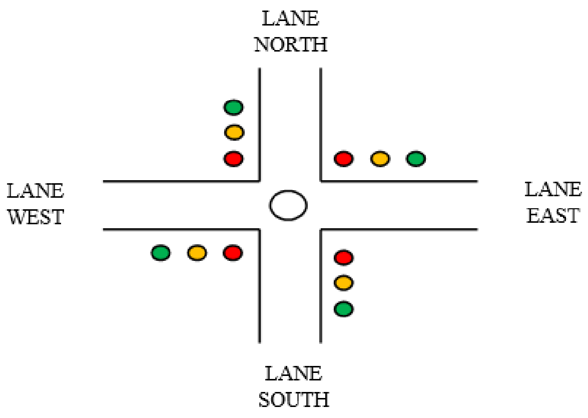
TRAFFIC LIGHT CONTROL

Traffic lights, which may also be known as stoplights, traffic lamps, traffic signals, signal lights, robots or semaphore, are signaling devices positioned at road intersections, pedestrian crossings and other locations to control competing flows of traffic.

ABOUT THE COLORS OF TRAFFIC LIGHT CONTROL

- Traffic lights alternate the right of way of road users by displaying lights of a standard color (red, yellow/amber, and green), using a universal color code (and a precise sequence to enable comprehension by those who are color blind).
- Illumination of the red signal prohibits any traffic from proceeding. Usually, the red light contains some orange in its hue, and the green light contains some blue, for the benefit of people with red-green color blindness, and "green" lights in many areas are in fact blue lenses on a yellow light (which together appear green).

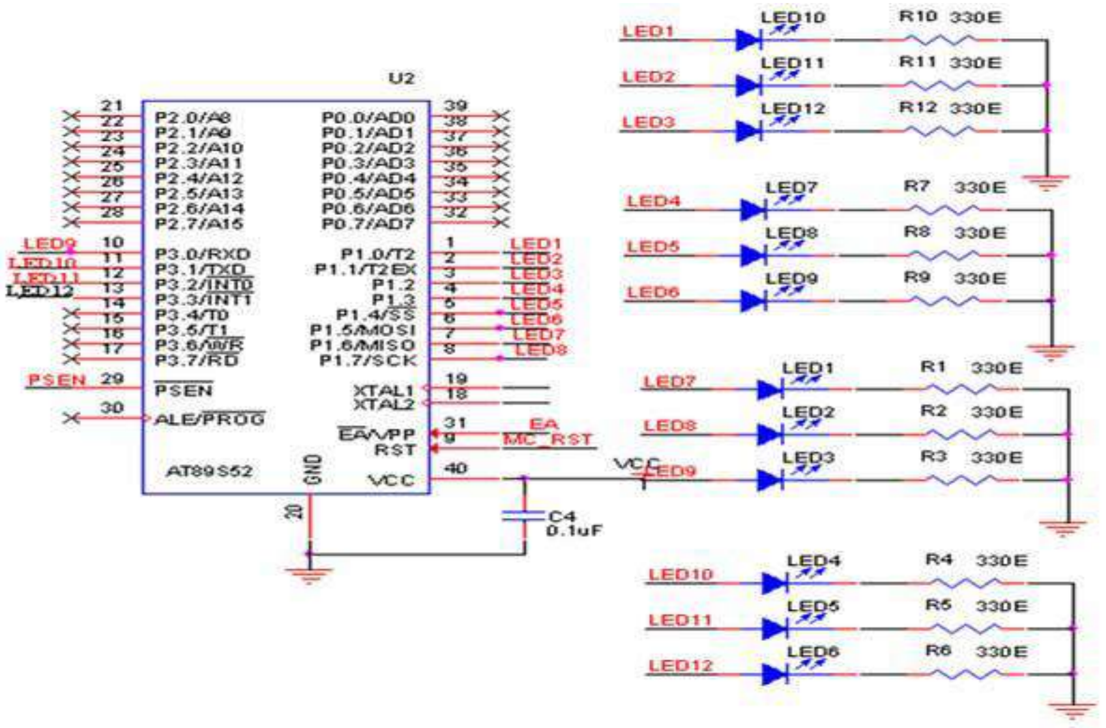




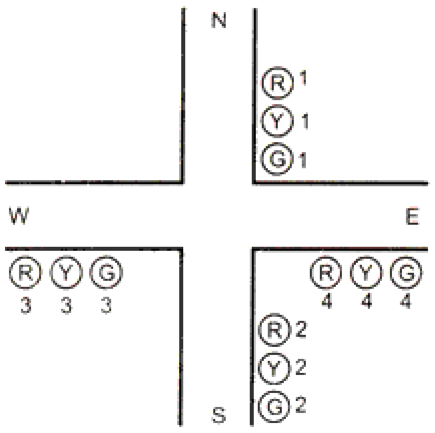
INTERFACING TRAFFIC LIGHT WITH 8085

The Traffic light controller section consists of 12 Nos. point LEDS are arranged by 4Lanes in Traffic light interface card. Each lane has Go (Green), Listen (Yellow) and Stop (Red) LED is being placed.

CIRCUIT DIAGRAM TO INTERFACE TRAFFIC LIGHT WITH 8085



HARDWARE FOR TRAFFIC LIGHT CONTROL



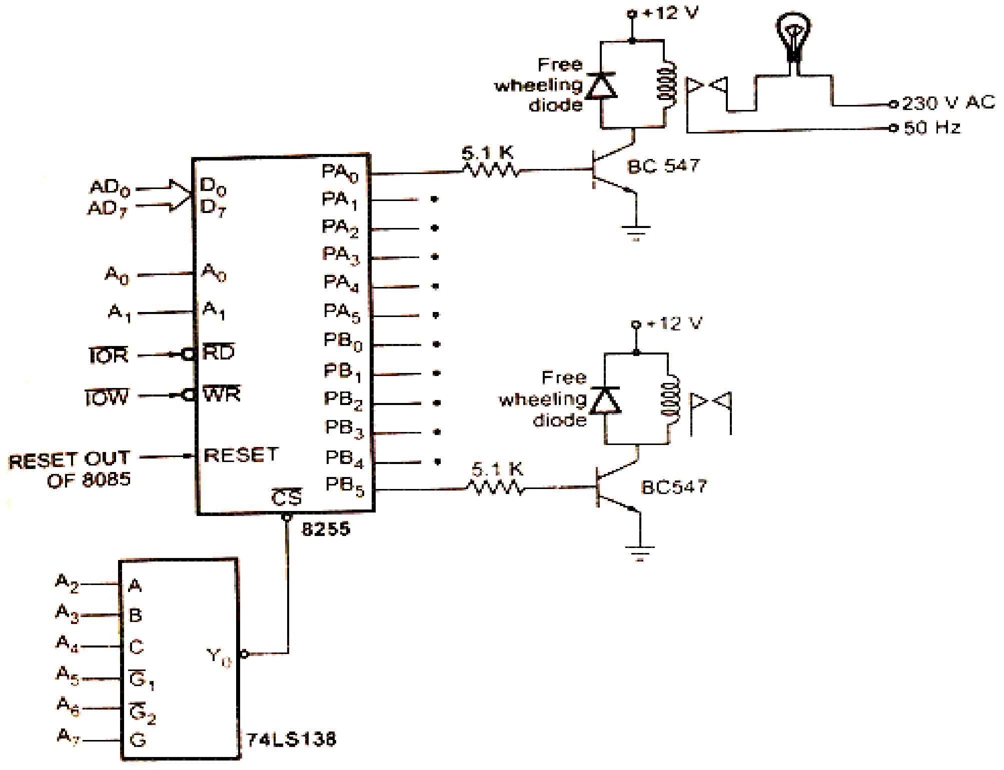
- The interfacing diagram to control 12 electric bulbs. Port A is used to control lights on N-S road and Port B is used to control lights on W-E road. Actual pin connections are listed in Table 1 below.

Pins	Light	Pins	Light
PA_0	R_1	PB_0	R_3
PA_1	Y_1	PB_1	Y_3
PA_2	G_1	PB_2	G_3
PA_3	R_2	PB_3	R_4
PA_4	Y_2	PB_4	Y_4
PA_5	G_2	PB_5	G_4

Table 1

- The electric bulbs are controlled by relays. The 8255 pins are used to control relay on-off action with the help of relay driver circuits. The driver circuit includes 12 transistors to drive 12 relays. Fig. also shows the interfacing of 8255

INTERFACING DIAGRAM:

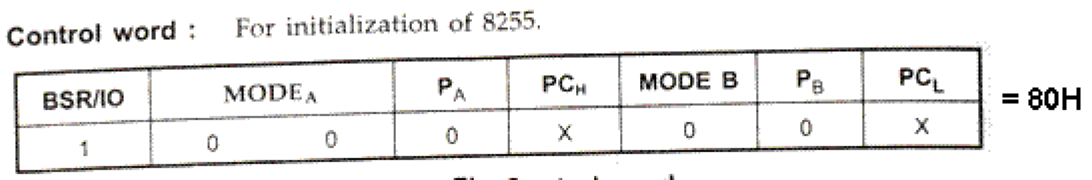


I/O MAP:

Ports / Control Register	Address lines A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0	Address
Port A	1 0 0 0 0 0 0 0	80H
Port B	1 0 0 0 0 0 0 1	81H
Port C	1 0 0 0 0 0 1 0	82H
Control Register	1 0 0 0 0 0 1 1	83H

Table 2

SOFTWARE FOR TRAFFIC LIGHT CONTROL:



Control word : For initialization of 8255.

BSR/IO	MODE$_A$		P_A	PC_H	MODE B	P_B	PC_L
1	0	0	0	X	0	0	X

= 80H

Fig. Control word

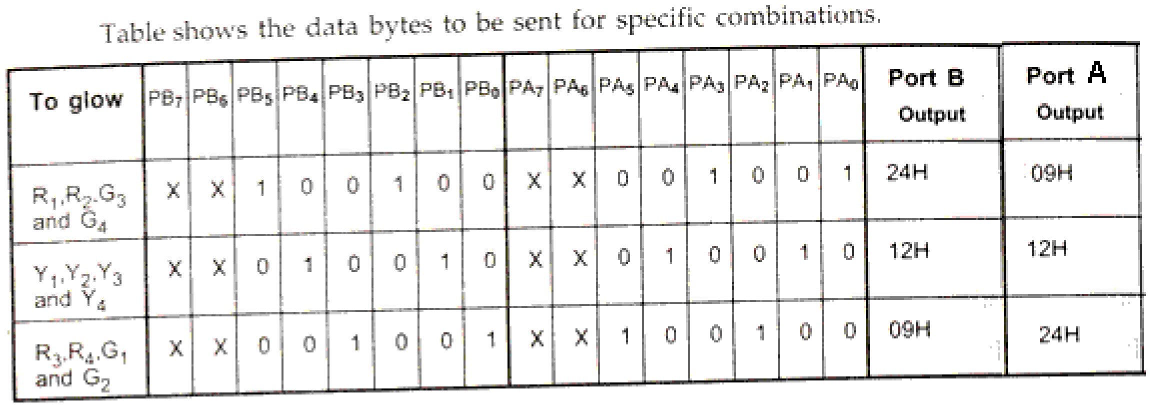


Table shows the data bytes to be sent for specific combinations.

To glow	PB_7	PB_6	PB_5	PB_4	PB_3	PB_2	PB_1	PB_0	PA_7	PA_6	PA_5	PA_4	PA_3	PA_2	PA_1	PA_0	Port B Output	Port A Output
R_1,R_2,G_3 and G_4	X	X	1	0	0	1	0	0	X	X	0	0	1	0	0	1	24H	09H
Y_1,Y_2,Y_3 and Y_4	X	X	0	1	0	0	1	0	X	X	0	1	0	0	1	0	12H	12H
R_3,R_4,G_1 and G_2	X	X	0	0	1	0	0	1	X	X	1	0	0	1	0	0	09H	24H

PROGRAM:

```
MVI A, 80H               : Initialize 8255, port A and port B
OUT 83H (CR)              : in output mode
START: MVI A, 09H
OUT 80H (PA)              : Send data on PA to glow R1 and R2
MVI A, 24H
OUT 81H (PB)               : Send data on PB to glow G3 and G4
MVI C, 28H                : Load multiplier count (40ıo) for delay
CALL DELAY                  : Call delay subroutine
MVI A, 12H
OUT (81H) PA                : Send data on Port A to glow Y1 and Y2
OUT (81H) PB                : Send data on port B to glow Y3 and Y4
MVI C, 0AH                : Load multiplier count (10ıo) for delay
CALL: DELAY                 : Call delay subroutine
MVI A, 24H
OUT (80H) PA                : Send data on port A to glow G1 and G2
MVI A, 09H
OUT (81H) PB                 : Send data on port B to glow R3 and R4
MVI C, 28H                 : Load multiplier count (40ıo) for delay
CALL DELAY                   : Call delay subroutine
MVI A, 12H
OUT PA                    : Send data on port A to glow Y1 and Y2
OUT PB                    : Send data on port B to glow Y3 and Y4
MVI C, 0AH                : Load multiplier count (10ıo) for delay
CALL DELAY                  : Call delay subroutine
JMP START
```

Delay Subroutine:

```
DELAY: LXI D, Count        : Load count to give 0.5 sec delay
BACK: DCX D                : Decrement counter
MOV A, D
ORA E                      : Check whether count is 0
JNZ BACK                   : If not zero, repeat
DCR C                      : Check if multiplier zero, otherwise repeat
JNZ DELAY
RET                        : Return to main program
```

4.10 DESIGN INTERFACE FOR STEPPER MOTOR CONTROL USING 8255:

STEPPER MOTOR:

- A stepper motor is a device that translates electrical pulses into mechanical movement in steps of fixed step angle.
- ✓ The stepper motor rotates in steps in response to the applied signals.
- ✓ It is mainly used for position control.
- ✓ It is used in disk drives, dot matrix printers, plotters and robotics and process control circuits.

Structure:

- Stepper motors have a permanent magnet called rotor (also called the shaft) surrounded by a stator.
- The most common stepper motors have four stator windings that are paired with a center-tap.
- This type of stepper motor is commonly referred to as a four-phase or unipolar stepper motor.
- The center tap allows a change of current direction in each of two coils when a winding is grounded, thereby resulting in a polarity change of the stator.

Interfacing:

- Even a small stepper motor require a current of 400 mA for its operation. But the ports of the microcontroller cannot source this much amount of current.
- If such a motor is directly connected to the microprocessor/microcontroller ports, the motor may draw large current from the ports and damage it.
- So a suitable driver circuit is used with the microprocessor/microcontroller to operate the motor.

Motor Driver Circuit (ULN2003)

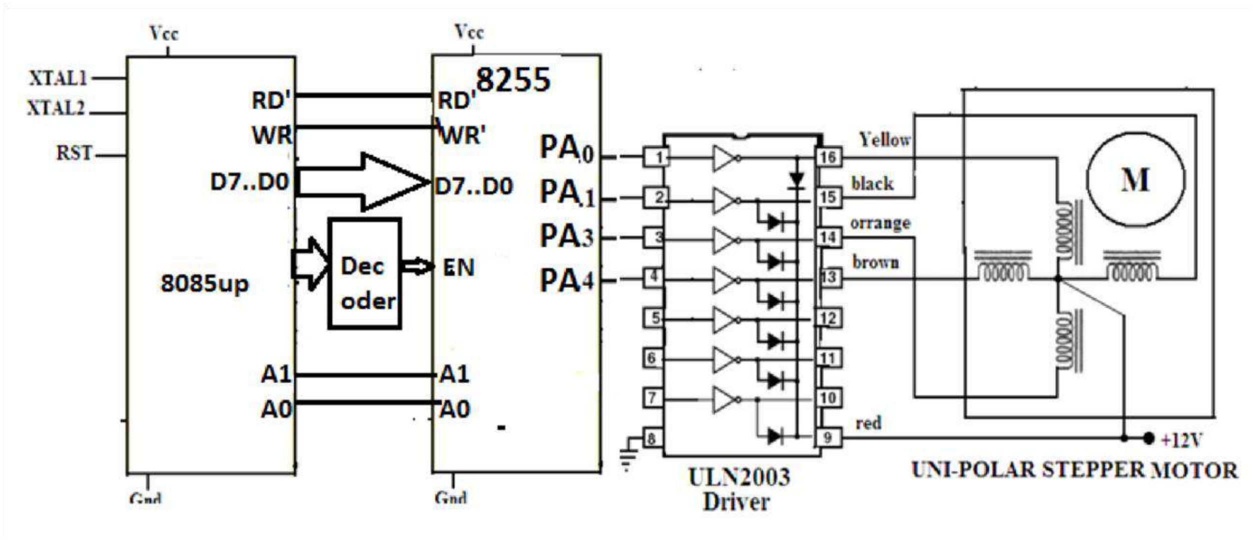
- Stepper motor driver circuits are available readily in the form of ICs.

- ULN2003 is one such driver IC which is a High-Voltage High-Current Darlington transistor array and can give a current of 500mA.
- This current is sufficient to drive a small stepper motor. Internally, it has protection diodes used to protect the motor from damage due to back emf and large eddy currents.
- So, this ULN2003 is used as a driver to interface the stepper motor to the microcontroller.

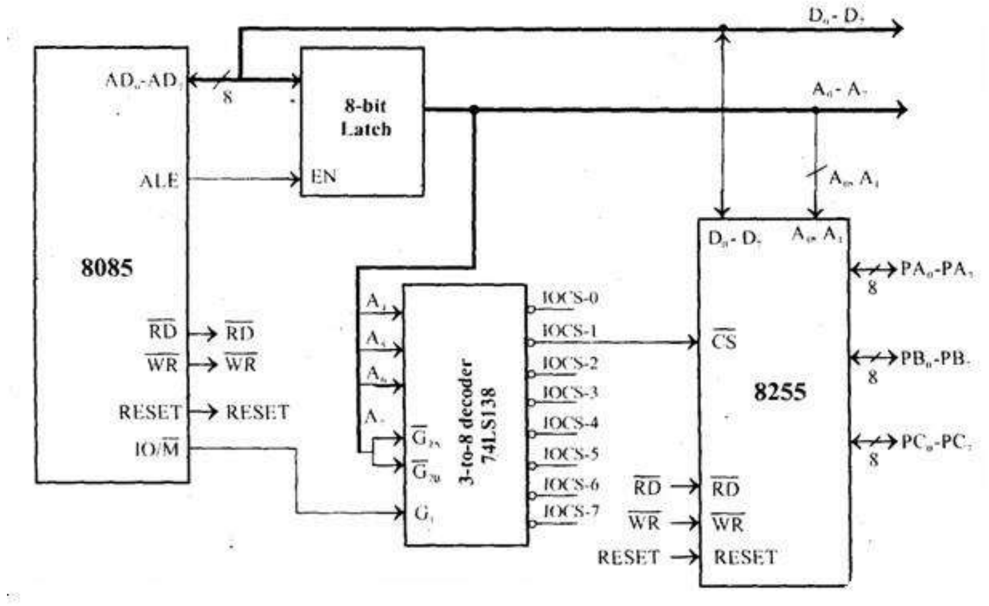
Operation:

- The important parameter of a stepper motor is the **step angle**.
- It is the minimum angle through which the motor rotates in response to each **excitation pulse**.
- In a four phase motor if there are 200 steps in one complete rotation then then the step angle is $360/200 = 1.8^0$.
- So to rotate the stepper motor we have to apply the excitation pulse. For this the controller should send a hexa decimal code through one of its ports.
- **The hex code mainly depends on the construction of the stepper motor**. So, all the stepper motors do not have the same Hex code for their rotation. (Refer the operation manual supplied by the manufacturer.)
- For example, let us consider the hex code for a stepper motor to rotate in clockwise direction is 77H, BBH, DDH and EEH. This hex code will be applied to the input terminals of the driver through the assembly language program. To rotate the stepper motor in anti-clockwise direction the same code is applied in the reverse order.

Stepper Motor interface- Schematic Diagram (for 8085):



Detailed Connection diagram between 8085 and 8255:



ASSEMBLY LANGUAGE PROGRAM (8085)

```
Main   : MVI                          ;   80H → Control word to
A, 80                                 configure PA,PB,PC in    O/P
OUT                                   ;  Write control
CWR_Address                           word in CWR
                                      of 8255
MVI A, 77                             ; Code for the
                                      Phase 1
OUT                                   ; sent to motor
PortA_Address                         via port A of
                                      8255              ;
CALL DELAY                            ;   Delay
                                      subroutine
MVI A,  BB                            ; Code for the
                                      Phase II
OUT                                   ; sent to motor
PortA_Address                         via port A of
                                      8255
CALL DELAY                            ;   Delay
                                      subroutine.
MVI A,  DD                            ; Code for the
                                      Phase III
OUT                                   ; sent to motor
PortA_Address                         via port A of
                                      8255;
CALL DELAY                            ; Delay
                                      subroutine
```

```
MVI A,  EE H            ;  Code for the Phase 1
OUT PortA_Address       ; sent to motor via port A of 8255        ;
CALL DELAY              ;  Delay subroutine
JMP MAIN                ; Keep the motor rotating continuously.
```

DELAY Subroutine

```
MVI C, FF               ;  Load C with FF -- Change it for the speed variation
LOOP1:  MVI D,FF        ; Load D with FF
LOOP2:  DCR D
JNZ  LOOP2
DCR C
JNZ LOOP1
RET                     ; Return to main program.
```

4.11 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 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.

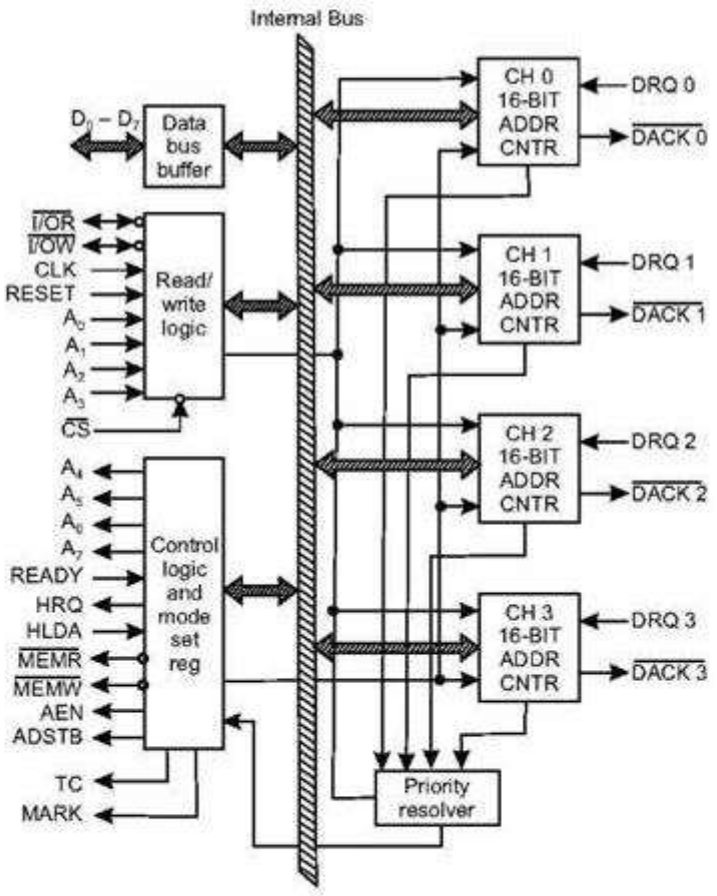
Features of 8257:

Here is a list of some of the prominent 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**.

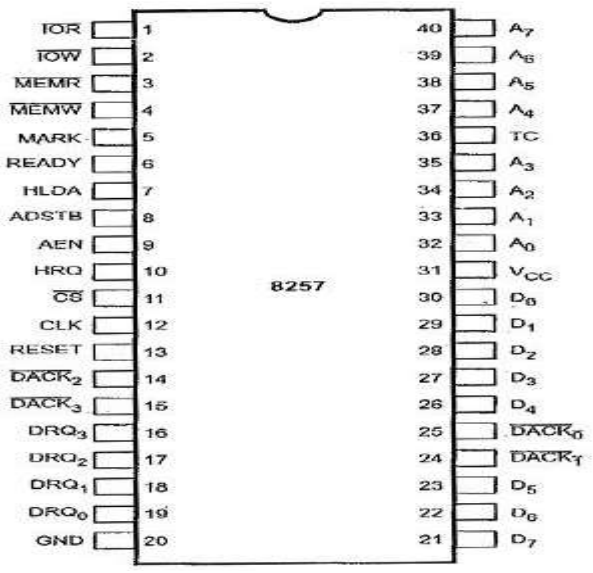
8257 Architecture:

The following image shows the architecture of 8257 –



8257 Pin Description:

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



DRQ$_0$–DRQ$_3$

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.

$DACK_0$ – $DACK_3$

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_0 – 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_0 - 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}

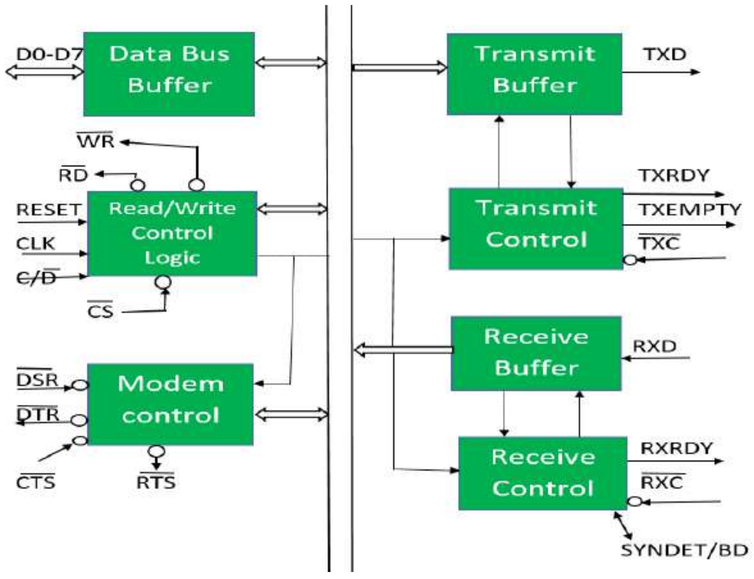
It is the power signal which is required for the operation of the circuit

4.12 USART: (8251)

8251 universal synchronous asynchronous receiver transmitter (USART) acts as a mediator between microprocessor and peripheral to transmit serial data into parallel form and vice versa.

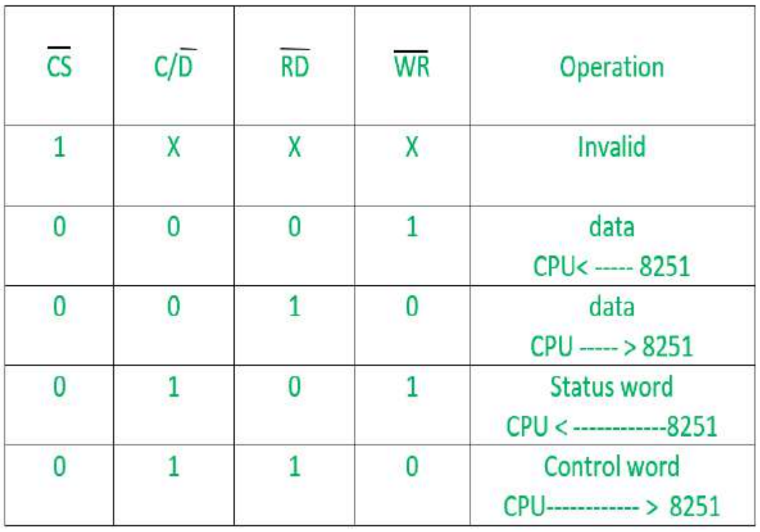
1. It takes data serially from peripheral (outside devices) and converts into parallel data.
2. After converting the data into parallel form, it transmits it to the CPU.
3. Similarly, it receives parallel data from microprocessor and converts it into serial form.
4. After converting data into serial form, it transmits it to outside device (peripheral).

Block Diagram of 8251 USART –



Data bus buffer:
This block helps in interfacing the internal data bus of 8251 to the system data bus. The data transmission is possible between 8251 and CPU by the data bus buffer block.

Read/Write control logic:
It is a control block for overall device. It controls the overall working by selecting the operation to be done. The operation selection depends upon input signals as:



$\overline{CS}$	C/$\overline{D}$	$\overline{RD}$	$\overline{WR}$	Operation
1	X	X	X	Invalid
0	0	0	1	data CPU< ----- 8251
0	0	1	0	data CPU -----> 8251
0	1	0	1	Status word CPU <------------8251
0	1	1	0	Control word CPU------------> 8251

In this way, this unit selects one of the three registers- data buffer register, control register, status register.

<u>**Modem control (modulator/demodulator):**</u>
A device converts analog signals to digital signals and vice-versa and helps

the computers to communicate over telephone lines or cable wires. The following are active-low pins of Modem.

- **DSR:** Data Set Ready signal is an input signal.
- **DTR:** Data terminal Ready is an output signal.
- **CTS:** It is an input signal which controls the data transmit circuit.
  RTS: It is an output signal which is used to set the status RTS.

<u>Transmit buffer:</u>

This block is used for parallel to serial converter that receives a parallel byte for conversion into serial signal and further transmission onto the common channel.

- **TXD:** It is an output signal, if its value is one, means transmitter will transmit the data.

<u>Transmit control:</u>

This block is used to control the data transmission with the help of following pins:

- **TXRDY:** It means transmitter is ready to transmit data character.
- **TXEMPTY:** An output signal which indicates that TXEMPTY pin has transmitted all the data characters and transmitter is empty now.
- **TXC:** An active-low input pin which controls the data transmission rate of transmitted data.

<u>Receive buffer:</u>

This block acts as a buffer for the received data.

- **RXD:** An input signal which receives the data.

<u>Receive control:</u>

This block controls the receiving data.

- **RXRDY:** An input signal indicates that it is ready to receive the data.
- **RXC:** An active-low input signal which controls the data transmission rate of received data.
- **SYNDET/BD:** An input or output terminal. External synchronous mode-input terminal and asynchronous mode-output terminal.