6.8. ADDRESS DECODING TECHNIQUES

In practice, all memory locations are not implemented. Hence, all the address lines of the microprocessor are not used by the memory module to select particular memory locations. Unused address lines are decoded to generate chip select signals. There are mainly two types of address decoding :

1. Fully address decoding.
2. Partial address decoding (linear address decoding).

6.8.1. Fully Address Decoding. In this system, all unused lines are decoded to generate chip select. Every unused line have a particular value either 1 or 0.

In fully address decoding system, each location has fixed address. During read, write, operation other locations are affected.

The size of the memory module is not reduced in case of this decoding. As the hardware is complicated, it is expensive.

6.8.2. Partial Address Decoding. In partial address decoding, all unused lines are not decoded to generate chip select. The value of undecoded address bit is don't care. The address of location is not fixed, each location may have one or more undecoded lines.

The greatest advantage of partial address decoding is that it gives us way to get the job done with a minimum hardware. Because fewer address lines are to be decoded, less hardware is needed.

On the other hand, its major disadvantage is that future expansion of memory is difficult, and generally requires redesign of the address decoder.

The partial decoding generates foldback memory, it means each register will have more than one address.

6.9. INTERFACING OF ROM

Interfacing of ROM. Read only memory (ROM) is a permanent and non-volatile memory, which is used to store the information that is not to be changed. Whenever we have to interface ROM/EPROM memory with microprocessor, then first see the various specifications of that memory. ROM memory has two active low control inputs :

(*i*) $\overline{CS}$ (Chip select) or $\overline{CE}$ (Chip enable)

(*ii*) $\overline{OE}$ (Output enable) or $\overline{RD}$ (Read bar).

The $\overline{CS}$ signal should be low only when the chip is to be selected or when a valid address appears on the address bus from the microprocessor.

The $\overline{OE}$ or $\overline{RD}$ signal places the data word on the output and this line should be driven by $\overline{MRD}$ control signal.

So whenever you have to interface ROM memory with microprocessor, then find,

(*i*) The number of address lines

(*ii*) The number of data lines

(*iii*) Prepare address decoding table for generating $\overline{CS}$

(*iv*) Connect address lines of chip with microprocessor address lines

(*v*) Connect data lines of chip with data lines of microprocessor

(*vi*) $\overline{CS}$ signal with the unused address lines

(*vii*) $\overline{OE}$ or $\overline{RD}$ signal with $\overline{MRD}$ signal of microprocessor.

SOLVED EXAMPLES

Example 6.1. *Connect 1K × 8 (1 K Byte) ROM with the system lines of 8085 system.*

Solution. In all the problems of memory mapping we have to draw following two figures :

(*a*) Demultiplexing of $AD_7 - AD_0$ pins.

(*b*) Generation of $\overline{MRD}$ and $\overline{MWR}$ from microprocessor control pins IO/$\overline{M}$, $\overline{RD}$, $\overline{WR}$.

1K × 8 ROM memory has following pins :

(*a*) 1K locations or rows means 10 address pins $A_9 - A_0$.

(*b*) 8 columns means 8 flip-flops in each location, so there will be 8-data pins.

(*c*) ROM has control signal only $\overline{RD}$.

(*d*) Common pins in all the memory IC are $\overline{CS}$, + 5 V and ground.

Address decoding table for generating $\overline{CS}$:

Memory IC 1K × 8 ROM

Hexadecimal	Binary Address															
	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
0000 H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
03FF H	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1

Decoded to generate $\overline{CS}$ — To A_9–A_0 of Memory IC

Connection of memory IC with the system bus/lines

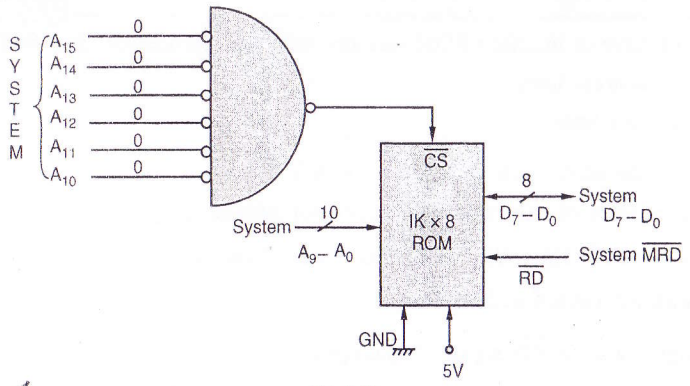


Fig. 6.10.

Example 6.2. *Connect 2K Byte EPROM with the system lines of 8085 MPU from address space 6000 H.*

Solution :

(*a*) Demultiplex $AD_7 - AD_0$ pins.

(*b*) $\overline{MRD}$ and $\overline{MWR}$ signals generation.

2K × 8 EPROM has following pins :

(*c*) (1) 2K locations means 11 address lines A_{10}–A_0.

(2) 8 columns means 8-data pins.

(3) EPROM has control pin $\overline{RD}$.

(4) $\overline{CS}$, + 5 V, grounds are common pins.

Memory IC	Hexadecimal Address	Binary Address															
		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
2K×8 EPROM	6000 H	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	67FF H	0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1
		Decoded to generate $\overline{CS}$					Directly to Memory IC										

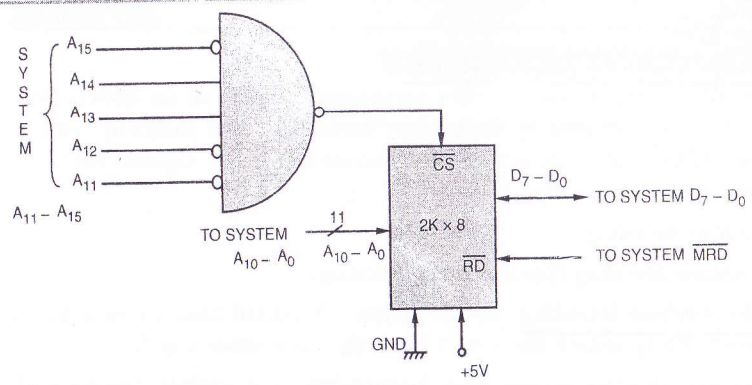


Fig. 6.11.

Example 6.3. *Connect 4K byte static RAM (SRAM) with microprocessor 8085. Assume starting address to be 4000 H.*

Solution. The given memory is SRAM of capacity 4K byte, the number of address lines will be 12 and number of data lines will be 8.

Memory IC	Hexadecimal Address	Binary Address															
		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
4K BYTE RAM	4000 H	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	4FFF H	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1
		Decoded to generate $\overline{CS}$					To A_{11}–A_0 of Memory IC										

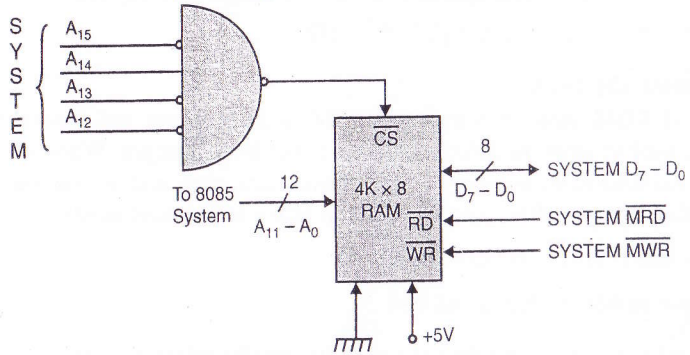


Fig. 6.12.

Example 6.4. (*a*) *Connect 4K bytes (4K × 8) EPROM with system lines of microprocessor 8085 starting from address C000 H.*

(*b*) *Connect 8K × 8 RAM with system lines of microprocessor 8085.*

Solution. *(a)* **4K × 8 EPROM :**

4K means 12 address lines $A_{11} - A_0$

8 = 8 data pins $D_7 - D_0$

Address Decoding Table

Memory IC	Hexadecimal Address	Binary Address															
		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
	C000 H	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	CFFF H	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1

Decoded to Generate $\overline{CS}$ — A_{11}–A_0 System Line to A_{11}–A_0 of Memory IC

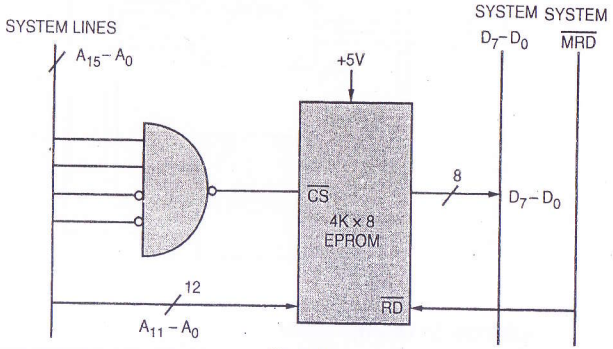


Fig. 6.13.

(b) **8K × 8 RAM :**

8K means 13 address lines $A_{12} - A_0$ RAM ⇒ Both $\overline{RD}$ and $\overline{WR}$

Address Decoding Table

Memory IC	Hexadecimal Address	Binary Address															
		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
8K× 8 RAM	4000 H	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	5FFF H	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1

Decoded to Generate $\overline{CS}$ — System A_{12}–A_0 Line to A_{12}–A_0 of Memory IC

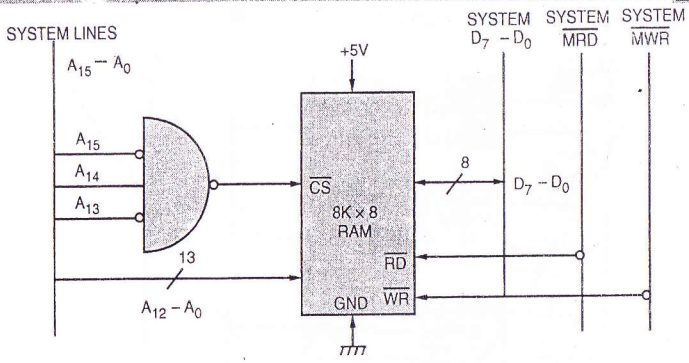


Fig. 6.14.

Example 6.5. *Connect 16 K bytes with the system lines of microprocessor 8085 performing partial address decoding i.e., use A_{15} line for generating $\overline{CS}$. The memory IC available is of EPROM and starting address is 8000 H.*

Solution. **Address Decoding Table**

Memory IC	Hexadecimal Address	Binary Address															
		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
16K× 8 EPROM	8000 H	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BFFF H	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Decoded to Generate $\overline{CS}$ — Not used for Decoding — System A_{13}–A_0 lines to A_{13}–A_0 of Memory IC

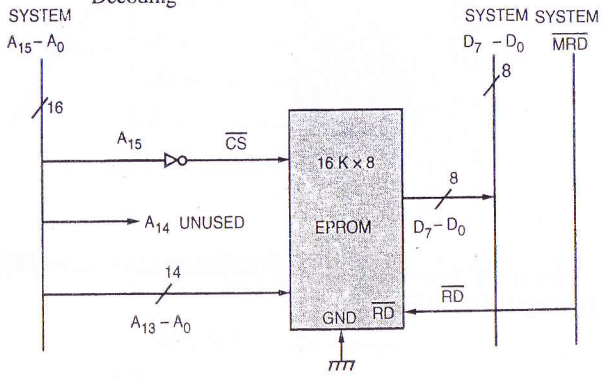


Fig. 6.15.

A_{14} can be either 0 or 1 so $\overline{CS} = 0$, even when A_{14} is 1 or 0, so each location has two address with 1 and 0.

To enable the decoder, we have to give $\overline{E_1} = \overline{E_2} = 0$ and $E_3 = 1$

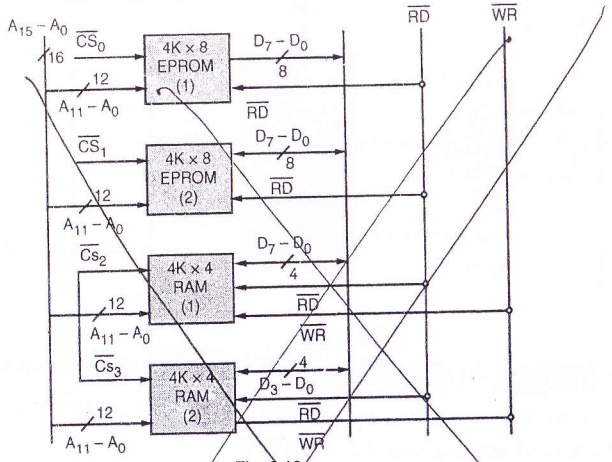


Fig. 6.18.

Connection of 3 to 8 decoder for generating $\overline{CS}$ of different IC's are shown below :

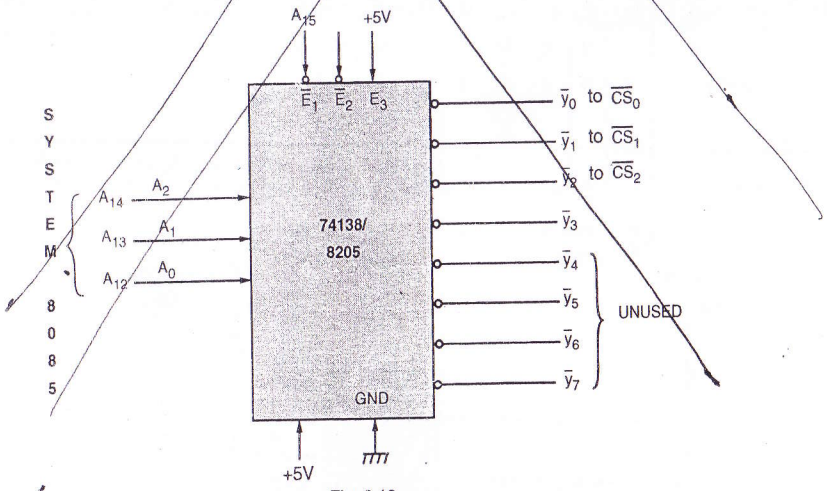


Fig. 6.19.

Example 6.8. *Connect 8K byte EPROM with the microprocessor 8085. The IC available is 2K × 8 EPROM (IC 2716).*

Solution. Total memory to be connected = 8K × 8 EPROM

IC available = 2K × 8 EPROM, so we have to connect four ICs in series.

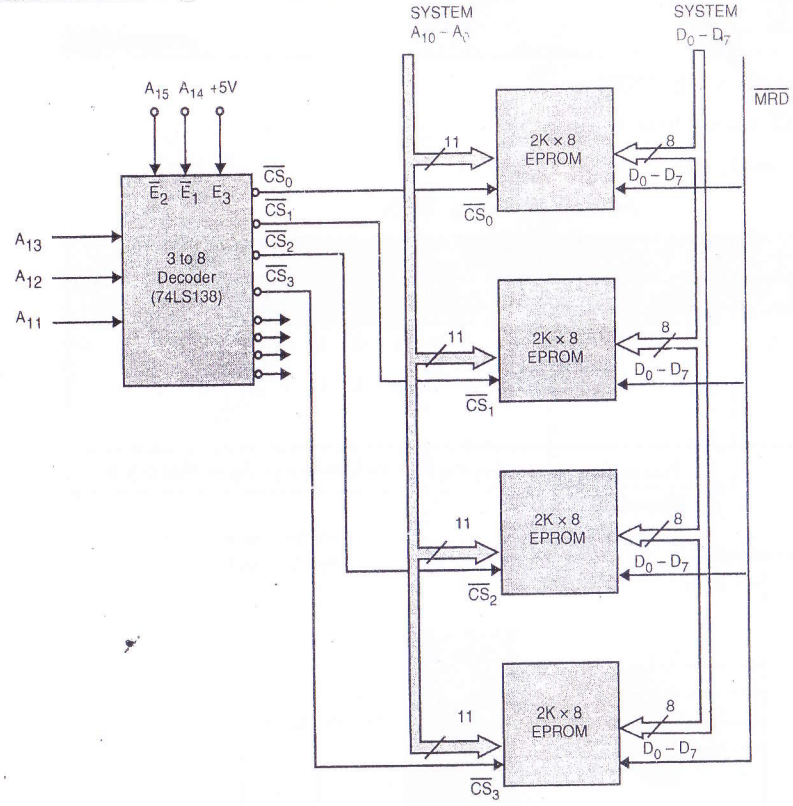


Fig. 6.20.

Address Decoding Table

S. No	Memory IC	Hexa-decimal Address	Binary Address														
			A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
1.	2K × 8	0000 H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	EPROM	07FF H	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
2.	2K × 8	0800 H	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	EPROM	0FFF H	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
3.	2K × 8	1000 H	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	EPROM	17FF H	0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1
4.	2K × 8	1800 H	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
	EPROM	IFFF H	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
			I/P to 3 – 8 Decoder		Decoded to generate $\overline{CS}$ logic			A_{10}–A_0 pins directly to Memory IC										

LOGIC DEVICES FOR INTERFACING :-

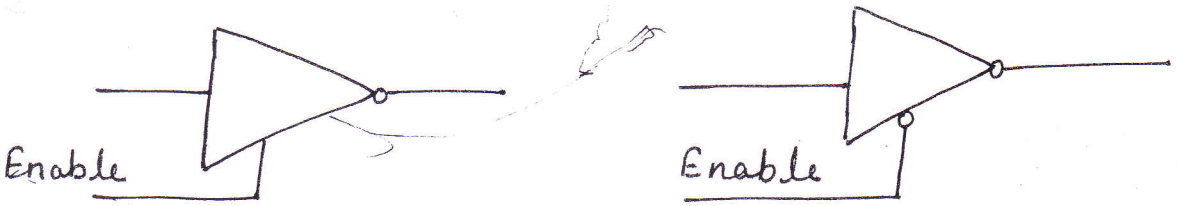
[1.] Tri-State Devices :- [When same bus lines are shared by several components.]

→ Tri-state logic devices has three states : logic 0, logic 1 & high impedance.

→ It has a third line called Enable.

→ When this line is activated, the tri-state device functions the same way as ordinary logic devices.

→ When this line is disabled, the logic device goes into the high impedance state.



a) Active High b) Active low

Tri-state inverter

a.) → When Enable is high – ckt funtion as ordinary inverte
→ " " " low – inverter stays in high impedan state.

b.) When Enable is high – inverter stays in high impedance state.

" " " low – function as ordinary inverter

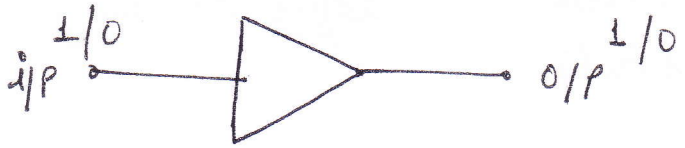
BUFFER (Driver)

→ Buffer is a logic circuit that amplifies the current or power.

→ The logic level of o/p is same as that of input.

→ logic 1 i/p → logic 1 o/p

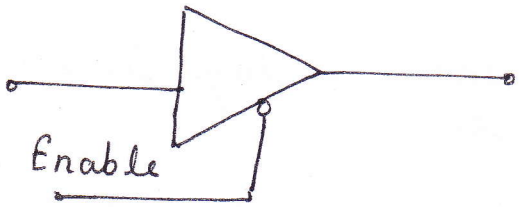
→ Buffer is used to increase the driving capability of a logic circuit. (also known as driver)



tristate buffer

→ When enable line is low, the ckt functions as a buffer.

→ Otherwise it stays in high impedance state.



Active low

EXAMPLES OF TRISTATE BUFFERS (74LS244)

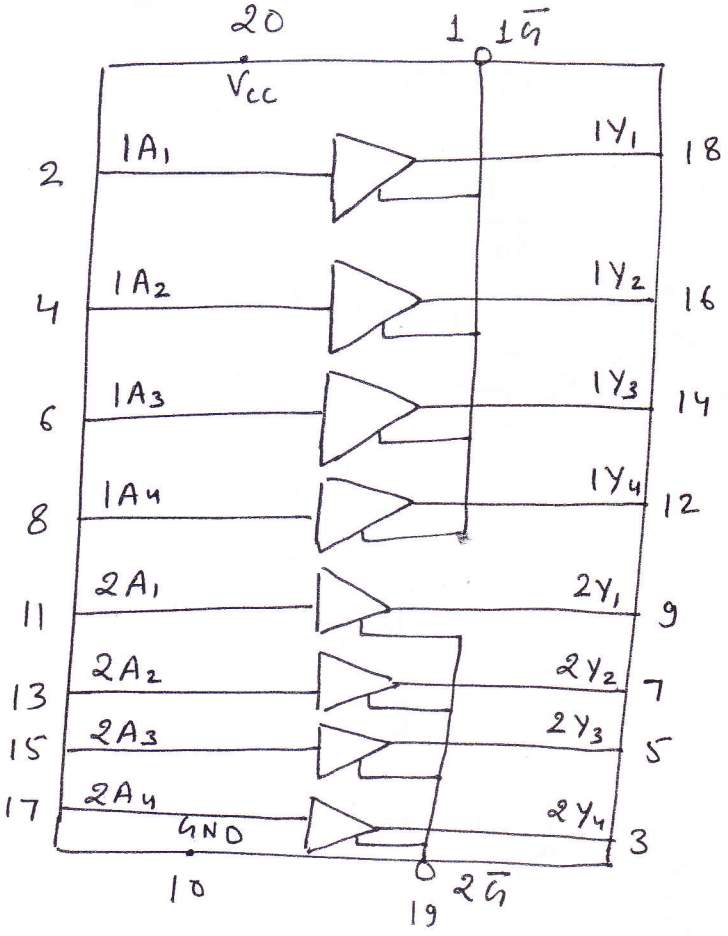
→ It is also known as a line driver or line receiver.

→ Used as a driver for the address bus in a bus-oriented system.

→ In fig, the buffers are controlled by two active low Enable lines ($1\bar{G}$ & $2\bar{G}$).

→ Each buffer sinking 24mA & sourcing -15mA of current.

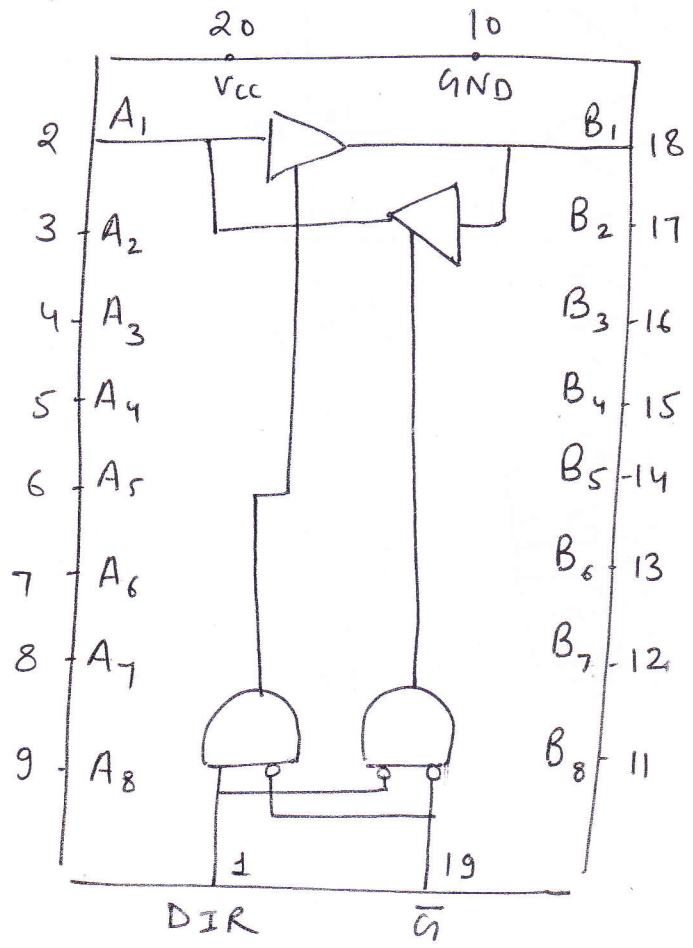
74LS244



BIDIRECTIONAL BUFFER :- (74LS245)

→ Bidirectional buffer also called Octal bus transceiver.
→ Used as a driver for the data bus.
→ It includes 16 bus drivers, eight for each direction, with tri-state output.
→ Direction of data flow is controlled by the pin DIR.
→ DIR at logic 1 – Data flow from A to B
" " " 0 – " " " B to A.
→ Enable signal ($\overline{G}$), which is active low.
→ Sink 24mA and source -15 mA of current.

74LS245



$\overline{G}$	DIR	Operation
0	0	B data to A Bus
0	1	A data to B "
1	X	Isolation

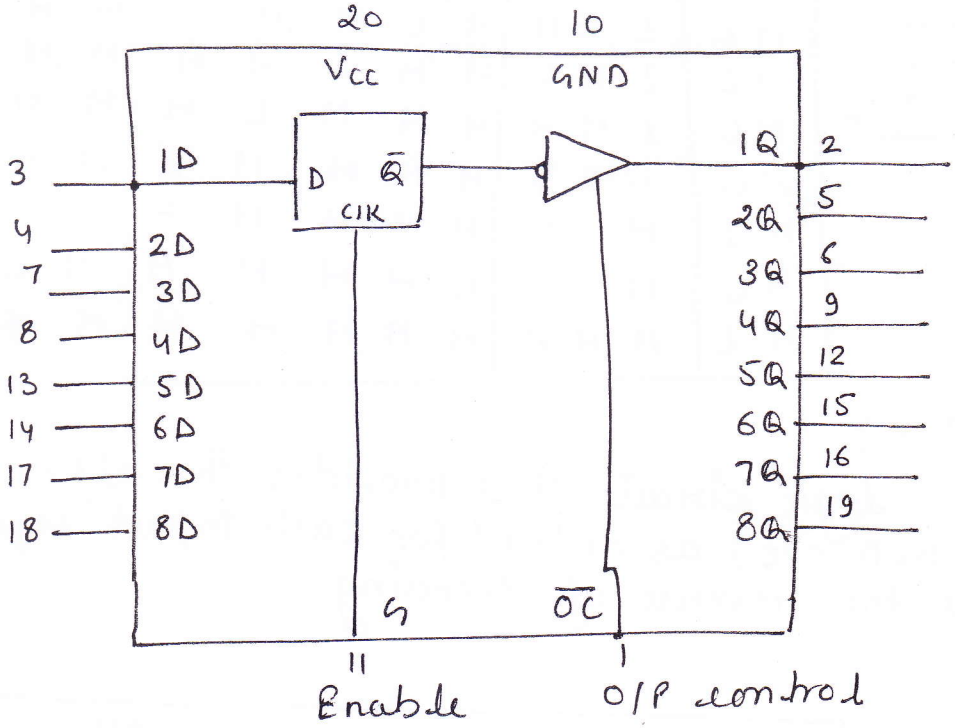
D flip-flop : latch & clocked :-

7475 D flip flop.

→ In this latch when the enable signal (G) is high, the O/P changes according to the I/P D.

Example of latches :-

Transparent latch (74LS373) :-

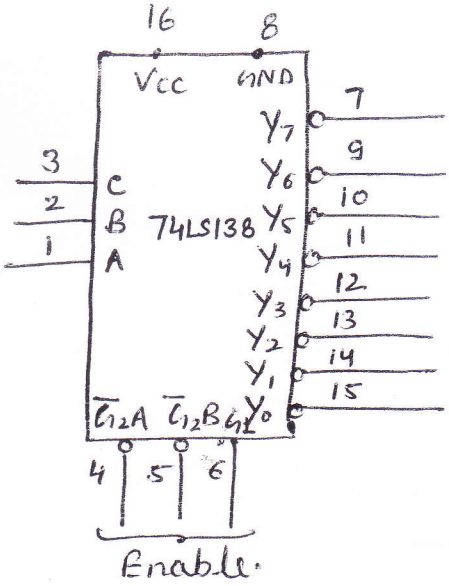


$\overline{OC}$	G	D	O/P
0	1	1	1
0	1	0	0
0	0	X	Q_0
1	X	X	Z

Decoder :- (n : 2^n)

→ It is a logic circuit that identifies each combination of the signals present at its input.

3 : 8 decoder



IC dia.

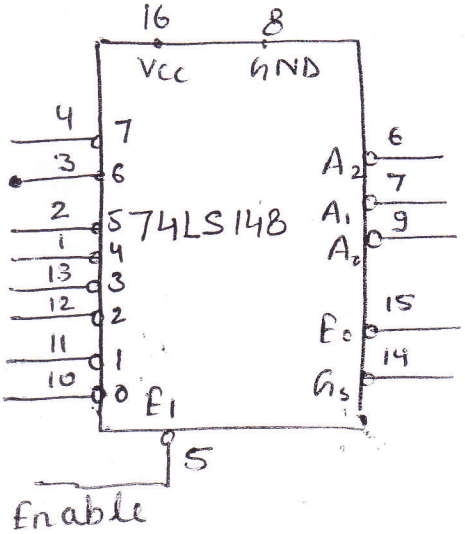
functional table

I/P's				O/P							
Enable		Select									
G_1	G_2	C	B	A	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

Encoder :- (2^n : n)

→ The encoder is a logic circuit that provides the appropriate code (binary, BCD etc) as output for each input signal.

→ The process is the reverse of decoding.



IC diagram

I/P's									O/P's	
E_1	0	1	2	3	4	5	6	7	A_2 A_1 A_0	G_S E_0
H	X	X	X	X	X	X	X	X	H H H	H H
L	H	H	H	H	H	H	H	H	H H H	H L
L	X	X	X	X	X	X	X	L	L L L	L H
L	X	X	X	X	X	X	L	H	L L H	L H
L	X	X	X	X	X	L	H	H	L H L	L H
L	X	X	X	X	L	H	H	H	L H H	L H
L	X	X	X	L	H	H	H	H	H L L	L H
L	X	X	L	H	H	H	H	H	H L H	L H
L	X	L	H	H	H	H	H	H	H H L	L H
L	L	H	H	H	H	H	H	H	H H H	L H

Instruction Word Size

Each instruction has two parts :-

Opcode :- task to be performed called the operation code (opcode)

Operand :- Data is to be operated called operand

→ According to word size or byte size classified into three groups →

(a) 1-byte instructions
(b) 2-byte "
(c) 3-byte instructions

One-Byte instructions :-

→ It includes the opcode & operand in the same byte.

	Opcode	~~Operand~~		
Copy the content of accumulator in reg. C	MOV	C, A	4F	0100 1111

opcode & operand in same byte

→ A mnemonic followed by a letter (or two letters) representing the reg. (such as A, B, C, D, E, H, L, M & SP) is a one-byte instr.

ex. a) MOV A, B
b) DCX SP
c) RRC

Two-byte instruction :-

A mnemonic followed by 8-bit is a two byte.

ex: (a) MVI A, 32H

(b) ADI 8-bit

3E	0011 1110	1st opcode
32	0011 0000	2nd operand

three-byte instruction :-

→ A mnemonic followed by 16-bit is a 3-byte inst.

ex: a) LXI B, 16 bit

b) LDA 2050H

0011 1010	3A	1st opcode
0101 0000	50	2nd Low order
0010 0000	20	3rd High order add

Addressing Modes :- (Way to achieve the task)

The different methods to select (address) the operands are called the addressing modes.

→ Immediate addressing
→ Register "
→ Direct "
→ Indirect "
→ Implied "

1.) Immediate addressing mode :-

→ the data (8/16 bit) is specified in the instruction itself.
→ Immediate addressing instructions are either 2 byte or 3 byte long.
→ Instruction containing the letter 'I'

ex:- MVI A, A0H

2.) Register Addressing mode :-

→ the source and destination operands are general purpose registers.
→ the reg. addressing instructions are generally of 1 byte i.e OPCODE only.

ex:- MOV D, B

3.) Direct Addressing Mode :-

→ The 16-bit address of the operand is given within the instruction itself.
→ the instructions in direct addressing mode are 3-byte instruction.

ex: LDA C200H

4.) Indirect Addressing Mode :-

→ the instructions reference the memory through a reg. pair i.e. the memory address where the operand is located is specified by the contents of a reg. pair.

ex: MOV A,M

5.) Implied Addressing or Implicit Addressing Mode :-

→ the implied mode of addressing does not require any operand.

→ the data is specified within the opcode itself.

→ 1-byte instruction.

ex: RAL

2.6 Timing Diagram :-

→ The graphical representation of any instruction is called "Timing diagram".

2.6.1 Instruction Cycle, Machine Cycle and T-states.

Instruction Cycle :-

→ "The total time required by microprocesser to fetching decoding and execution of a single instruction is called as instruction cycle"

→ $IC = FC + EC$

IC = time required for instruction cycle
FC = time required for fetch ~~execution~~ cycle
EC = time required for execution cycle.

→ One instruction cycle consist of either one or more than one machine cycles.

Machine Cycle :-

→ "The time required by the microprocessor to ~~complete~~ perform the operation from memory or I/O device is called machine cycle"

→ Operations are :-
- Opcode fetch
- Memory Read
- Memory Write
- I/O read
- I/O write

→ One machine cycle consist of more than one T-state

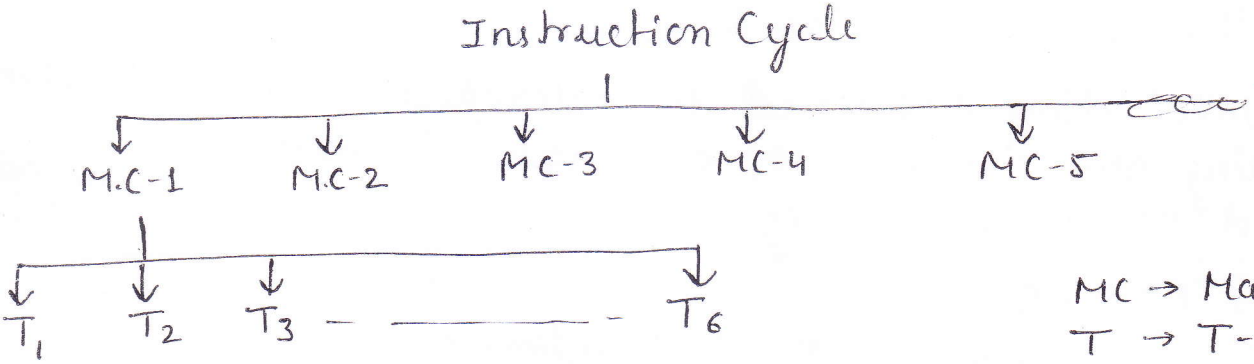
Machine cycle	Status			Control		
	IO/$\overline{M}$	S_1	S_0	$\overline{RD}$	$\overline{WR}$	INTA
Opcode Fetch	0	1	1	0	1	1
Memory Read	0	1	0	0	1	1
Memory Write	0	0	1	1	0	1
I/O Read	1	1	0	0	1	1
I/O Write	1	0	1	1	0	1
INTR Acknowledge	1	1	1	1	1	0
Bus Idle	0	0	0	1	1	1

2 or 5 Machine Cycles

T-state :-

→ Microprocessor performs an operation in a specific clock cycles. Each clock cycle is called as T-state.

→ Each machine cycles consists of 3 to 6 clock periods / cycles or T-state.



2.6.2

Types of Signals

1.) **Clock Signal :-**

→ 8085 divides the clock frequency provided at X_1 and X_2 inputs by 2, which is called operating frequency.

→ Clock contains some rise time T_R and fall time T_F as shown in fig.

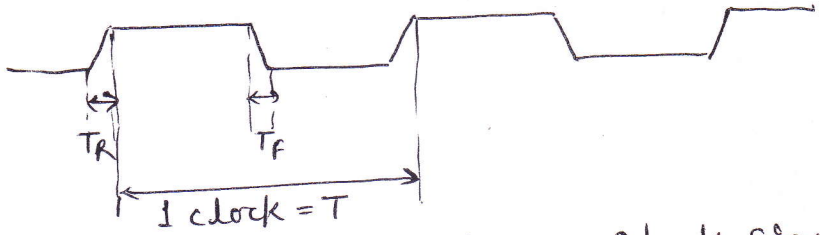


Fig-2.6 Clock Signal

2.) **Single Signal :-**

→ Single signal is represented by a single line.

→ This signal may be either zero or one.

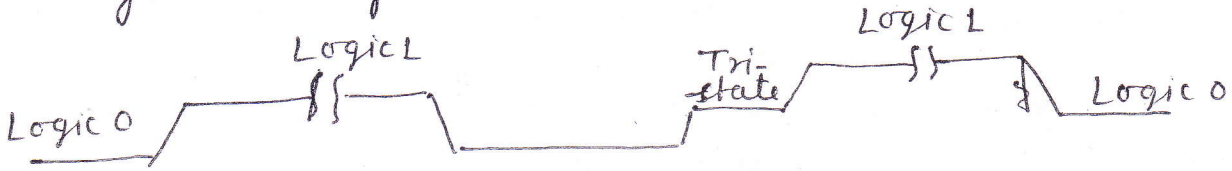
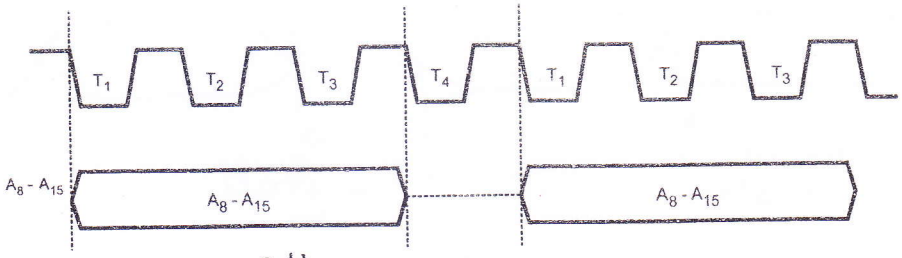


Fig 2.7 Single Signal

6) A_8-A_{15} (Higher byte address) :-

→ The higher byte of address is available on the A_8-A_{15} bus during T_1, T_2 and T_3-state of each machine cycle



2.11

Fig. 2.25 Higher byte address on A8-A15

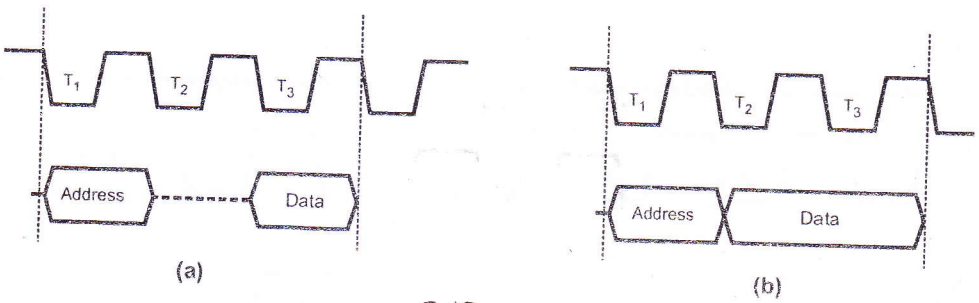
7) D_0-D_7 (Data Bus)

→ Data bus consist of 8 data lines D_0-D_7.

→ Data bus is transferred during T_2 and T_3 states.

→ During Read cycle, data will appear on the data bus during later part of T_2-state as shown in fig 2.12 (a).

→ During Write cycle, data will appear on the data bus at the beginning of the T_2-state as shown in fig 2.12 (b)



2.12

Fig. 2.24 Data bus

3.) Group signal :-

→ Group of signal is called a bus.

→ Bus consists of a number of single line.

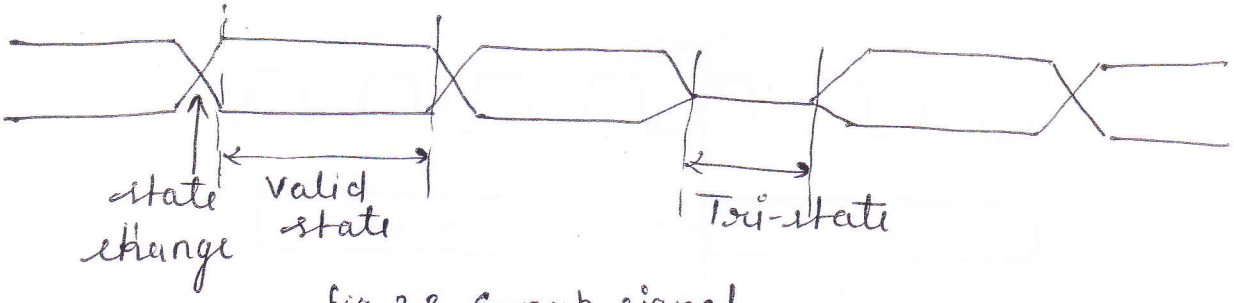


Fig-2.8 Group signal

4.) ALE :-

→ This signal is active high signal.

→ It is activated in the beginning of the T_1-state of each machine cycle.

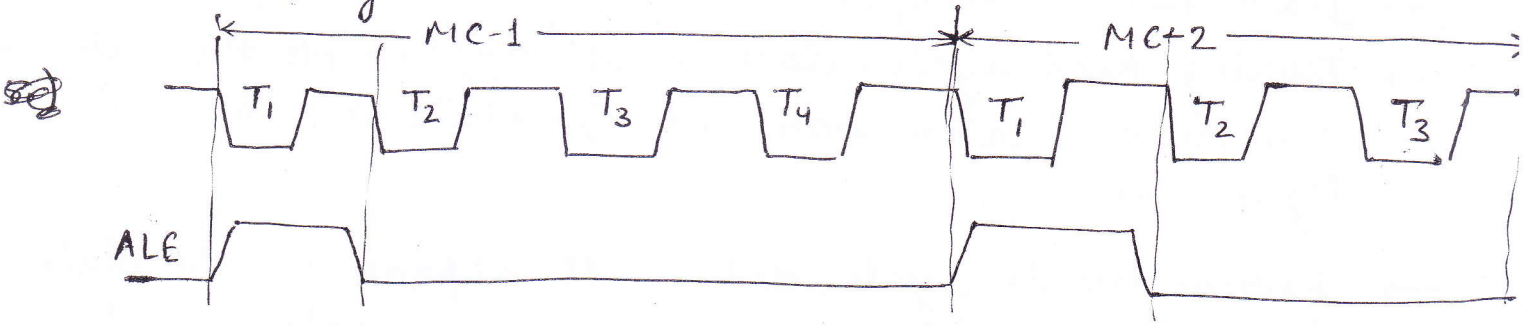
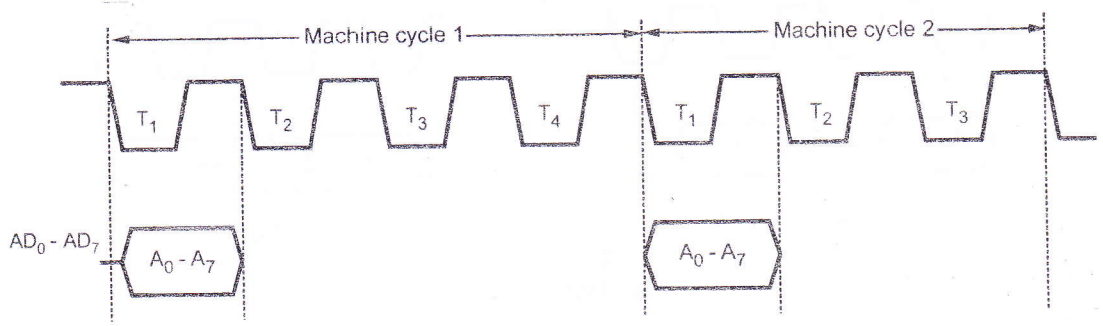


Fig 2.9 ALE signal

5.) A_0-A_7 (Lower byte address) :-

→ Lower address is obtained from AD_0-AD_7 line.

→ Lower address is applied during 1st T_1 state



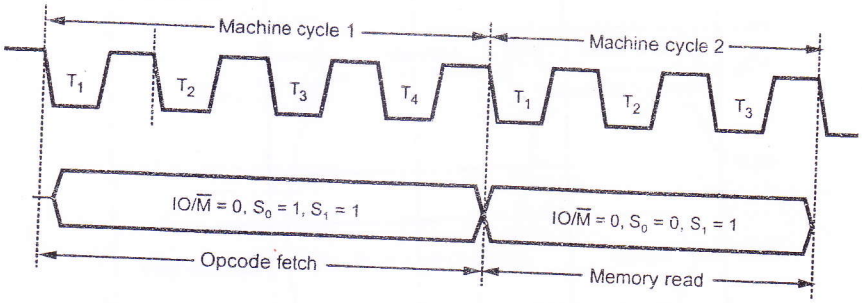
2.10

Fig. 2.10 Lower address on the multiplexed bus

8.) $IO/\overline{M}, S_0, S_1$:- (Status Signals)

→ These signals are called status signals.

→ They are activated at the beginning of T_1-state of each machine cycle and remain active till the end of the machine cycle.



2.13

Fig. ~~2.26~~ Status signals

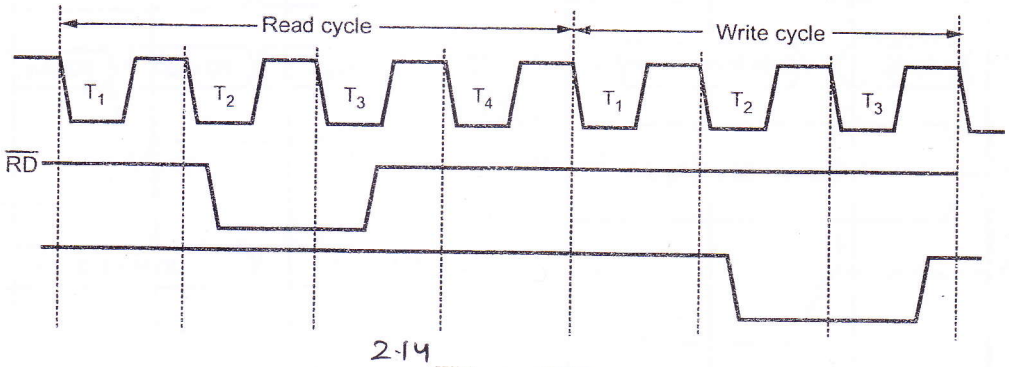
9.) $\overline{RD}$ and $\overline{WR}$:-

→ Decide the direction of the data transfer.

→ When $\overline{RD}=0$ data are transferred from memory or I/O delive to the microprocessor.

→ When $\overline{WR}=0$ data is transferred from microprocessor to the memory or I/O device.

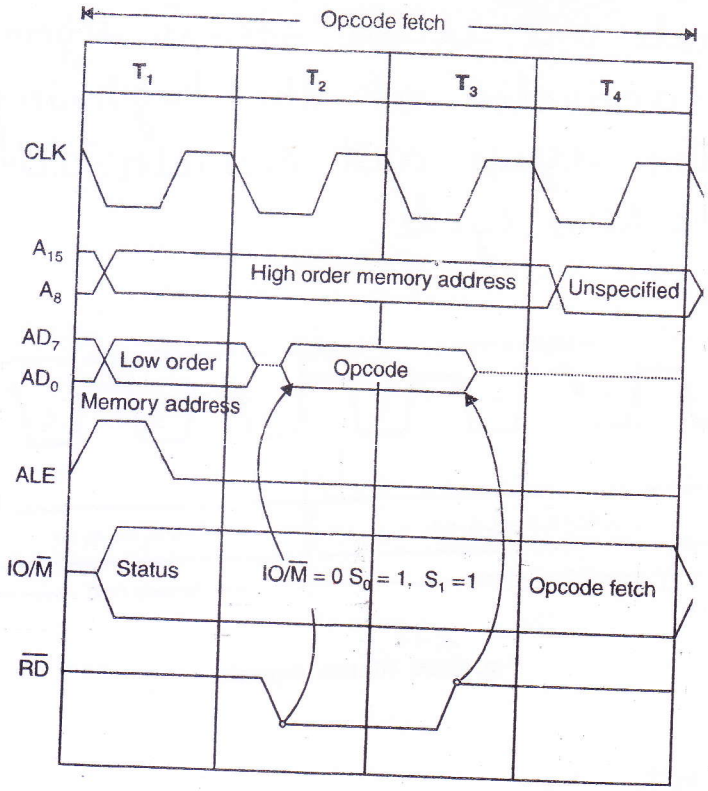
→ Data transferred b/w T_2 and T_3 states.



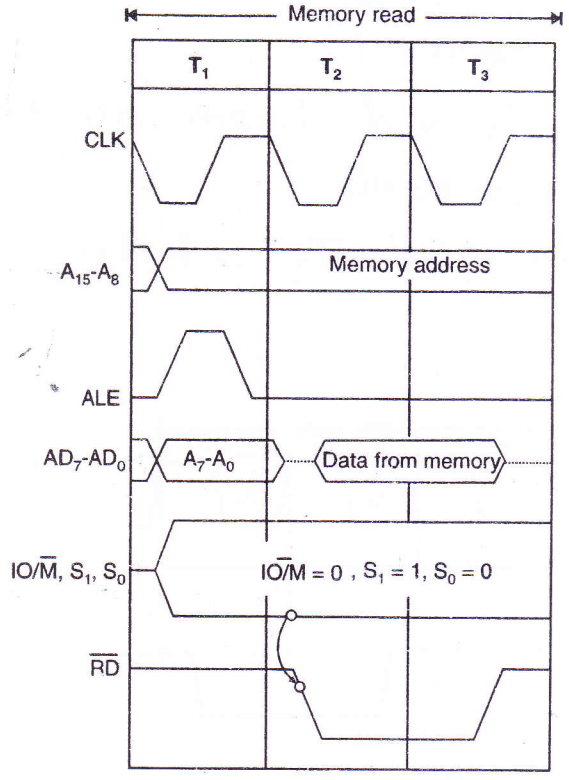
2.14

Fig. ~~2.27~~ $\overline{RD}$ and $\overline{WR}$ signals

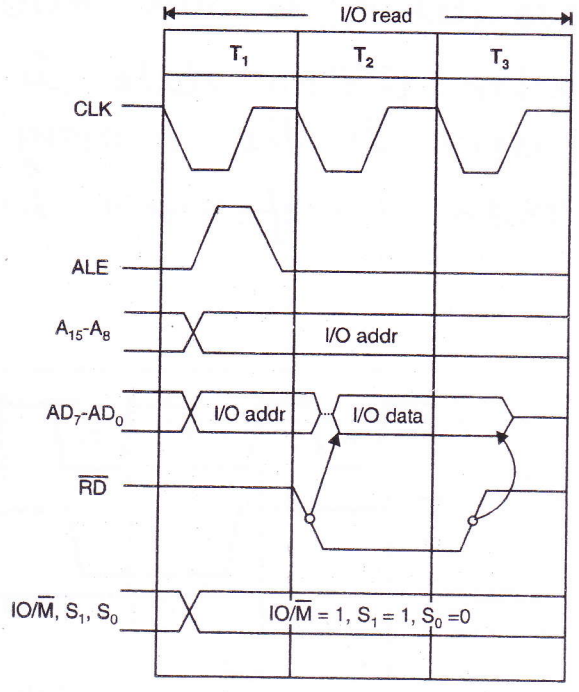
1) Opcode fetch :-



2.) Memory Read :-



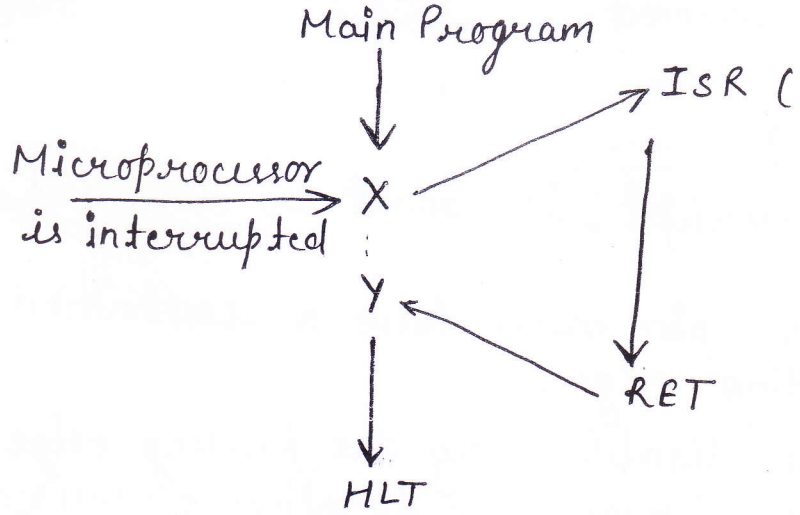
3.) I/O Read :-



Concepts of Interrupts :-

→ When microprocessor executes any program main progrom, if microprocessor is interrupted, then it will branch to sub program called Interrupt serivce routine (ISR)

→ MP will execute ISR and when 'RET' instruction comes MP returns back to main progrom instruction Y.



There are two types of Interrupt :

1.) Software Interrupts
2.) Hardward interrupts

[1] SOFTWARE INTERRUPTS :- (Right)

RST n [Restart from address (8xn) H] n = 0 to 7

RST 0	(8X0 = 0)	= 0000H
RST 1	(8X1 = 8)	= 0008H
RST 2	(8X2 = 16)	= 0010H
RST 3	(8X3 = 24)	= 0018H
RST 4	(8X4 = 32)	= 0020H
RST 5	(8X5 = 40)	= 0028H
RST 6	(8X6 = 48)	= 0030H
RST 7	(8X7 = 56)	= 0038H

[2] Hardware Interrupts :- (five)

Priority order				Address Location
1st	TRAP (or RST 4.5)	vectored	Non-maskable	0024H
2nd	RST 7.5	vectored	Maskable	003CH
3rd	RST 6.5	vectored	Maskable	0034H
4th	RST 5.5	vectored	Maskable	002CH
5th	INTR	non vectored	Maskable	Response to INTA

TRAP :- (or RST 4.5)

→ Non-maskable interrupt, it cannot be disabled by an instruction.
→ The s/g on the TRAP pin must have a sustained HIGH level with a leading edge.
→ The interrupt is disabled by the falling edge of the signal on the pin. (branch to location 0024H)

RST 7.5 :-

→ Maskable interrupt, it can be enabled or disabled using SIM instruction.
→ 8085 responds to RST 7.5 when the signal on RST 7.5 pin has a leading edge. (branch to location 003CH)

RST 6.5 :-

→ Maskable interrupt. It can be enabled or disabled with the SIM instruction and is HIGH level sensitive. (branch to location 0034H)

RST 5.5 :-

→ Maskable interrupt. It can be enabled or disabled by the SIM instruction and is HIGH level sensitive. (branch to location 002CH)

INTR :- → Maskable or hand shake interrupt & HIGH level sensitive.
→ The s/g on INTR pin is HIGH, 8085 completes execution of current instruction and generates interrupt acknowledge (INTA)

Classification of 8085 instructions :-

1.) **Data transfer instructions :-** data transfer instructions are used to move data between processor registers, memory and I/o devices.

ex: MOV , LDA , STA etc.

2.) **Arithmetic group :-** Arithmetic instructions are used to perform arithmetic operation like addition, subtraction, increment / decrement etc.

ex: ADD, SUB, INC, DCR etc.

3.) **Logical group :-** Logical instructions are used to perform logical operations like Rotate, Compare, AND, OR, complement etc.

ex:- RLC, RRC, CMP , AND, OR etc.

4.) **Branching group :-** Branching instructions are used to transfer the execution control.

ex: Jump, Call, Return etc.

5.) **Stack & Machine Control Group :-** Stack instructions are used to read / write stack memory.
Machine control instructions are used to control the system like enabling / disabling interrupts.

ex: PUSH, POP, HLT, IN, OUT etc.