U-4

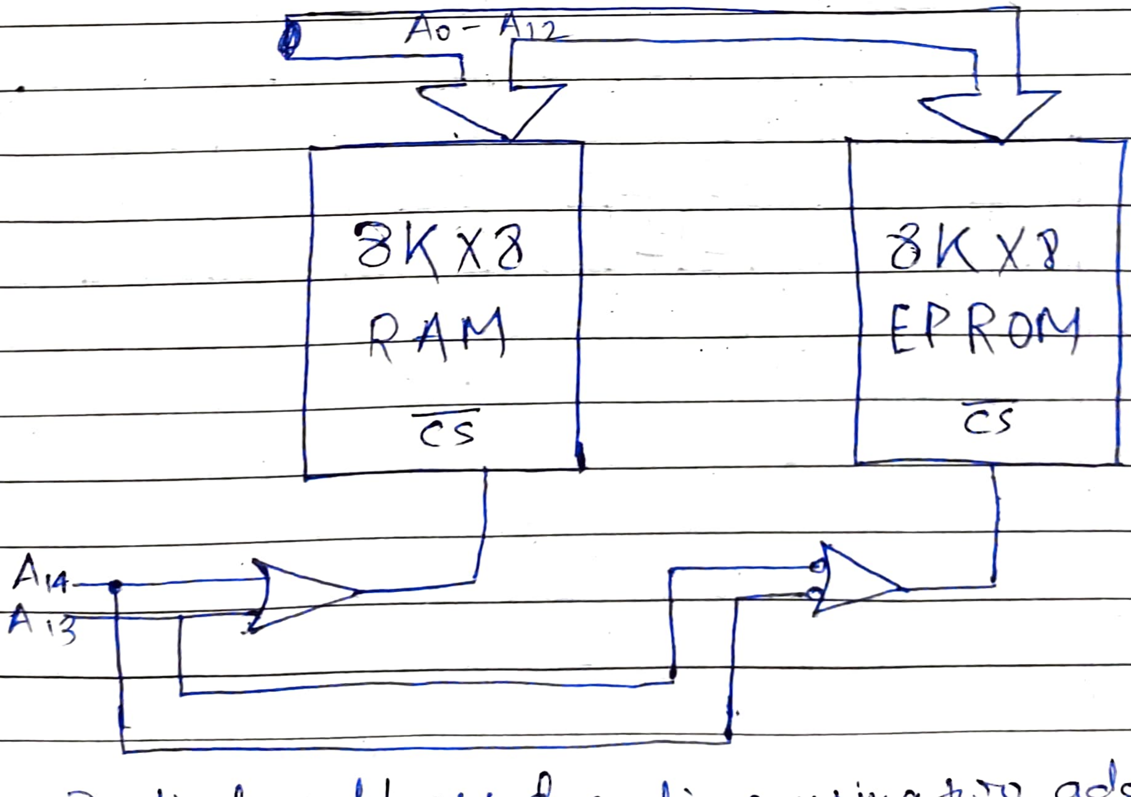
Ch - 1

Q.) ~~How many address lines does a~~
~~256~~

Q.) How many address & data lines are needed for the memory chips with following organisation?
256 X 4

→ It means it needs 4 data lines and 8 address lines as $2^8 = 256$.

Q.) Draw a decoding circuit using partial decoding for a RAM and EPROM, each of size 8K X 8. For decoding, use only the address lines A_{13} and A_{14}. What is the size of it's fold back memory



Partial address decoding using two address lines

Soln :

For the RAM, it is mandatory to have A_{13} & A_{14} to be 00. Since 5 address ~~are~~ lines are don't cares, there are 32 different ~~ad~~ addresses with which each location can be accessed.

A_{19}	A_{18}	A_{17}	A_{16}	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
X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
X	X	X	X	X	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1

For the EPROM, it is mandatory to have A_{13} & A_{14} to be 11. The address can vary as shown below. Since 5 lines are don't cares, there are 32 different addresses with which each location can be accessed.

A_{19}	A_{18}	A_{17}	A_{16}	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
X	X	X	X	X	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
X	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Ch - 2

Q.) Interrupt response of 8086.

→ The sequence of steps:

i) The flag register is pushed onto the stack.

ii) The interrupt flag is disabled (IF = 0)

iii) The trap flag is disabled (TP = 0)

iv) The CS register is pushed on to the stack.

v) The IP register is pushed on to the stack.

vi) Control is transferred to the location in which the "Interrupt service routine" (ISR) is stored. This is far jump

vii) The program corresponding to ISR is executed. The last instruction in the ISR will be IRET.

viii) Then IP is popped off the stack.

ix) CS is popped off the stack.

x) The flag register is popped off the stack.

xi) Control is returned to the point at which it had left off.

Q.) Explain the terms 'interrupt service routine' and interrupt vector'.

→ ISR :
when an interrupt occurs, the processor suspends the execution of it's current task and takes on another task as required by the Interrupt source. This program or routine is ~~a~~ called 'ISR'.

→ Interrupt vector :
~~For ISR to be available and~~
ISR should be available in memory & must be accessed on the occurence of the specific interrupt. For that, address of ISR must be obtained. The address of ISR is called it's 'interrupt vector'.

Q. Find the address (in the IVT) of the interrupt vector of INT 61h. Find the physical address of the ISR corresponding to this interrupt if the vector is 0F00 : 9872.

→ Type no. of interrupt = 61H = 97 in decimal
Address of interrupt vector is 97×4 = 388 = 184H

The interrupt vector is to be stored in the IVT in location 0000:0184 onwards

For ISR, CS value = 0F00H
IP value = 9872H

	0FH
0186H	00
	98H
0184H	72H

Q.) Discuss the five different Intel dedicated interrupt types directly related to CPU operations.

→

i) INT 0 (DIVIDE BY ZERO ERROR):

The interrupt with type 0 is dedicated to the 'divide by zero' error. On divison, if the quotient register is not large enough to contain the quotient, this interrupt is generated automatically. Type 0 means, in interrupt vector table it is available at 0000:0000.

ii) INT 1 (Single stepping):

This type number is dedicated for 'single stepping' or 'trace'. Single stepping is an important idea in debugging. During logical debugging we would like to stop after the execution of each instruction and check the content of registers, memory etc. We perform the action of 'trace' this way. Intel has provided the 'Trap' flag memory for this, and this flag has to be set to let this happen.

iii) INT 2 (Non Maskable Interrupt):
This interrupt corresponds to the vector of the hardware interrupt NMI. When an interrupt is received on the pin NMI of the processor, a type 2 interrupt occurs.

iv) INT 3 (BREAKPOINT Interrupt):
Breakpoint interrupt is useful for debugging. We need to set breakpoints ~~and~~ check the contents of registers and memory after executing instructions upto the breakpoint.

v) INT 4 (Overflow interrupt):
This interrupt corresponds to the overflow flag. If the overflow flag is set, this interrupt occurs, but not automatically. An instruction INTO must be written after the program segment which is likely to cause the overflow flag to be set.

Ex:

```
        MOV AL, NUM1
        ADD AL, NUM2
```

~~last line of this~~:

```
        INTO    ; interrupt on overflow
```

last line of this program segment can pass control to ISR written for INT 3, if OF is set. Otherwise INTO acts as a NOP.