

Module 3

- Stack Structure of 8086

- SP is 16-bit register that points to top of stack
- Lies in the SS
- SS may have a memory block of a max of 64 Kbyte locations
- Stack seg register (SS) contains the base address of the stack seg in the memory
- SS and SP together address the stack-top

- Let SS =5000H, SP= 2050H .Find current stack top address?

SS \Rightarrow 5000 H

SP \Rightarrow 2050 H

SS \Rightarrow 0101 0000 0000 0000

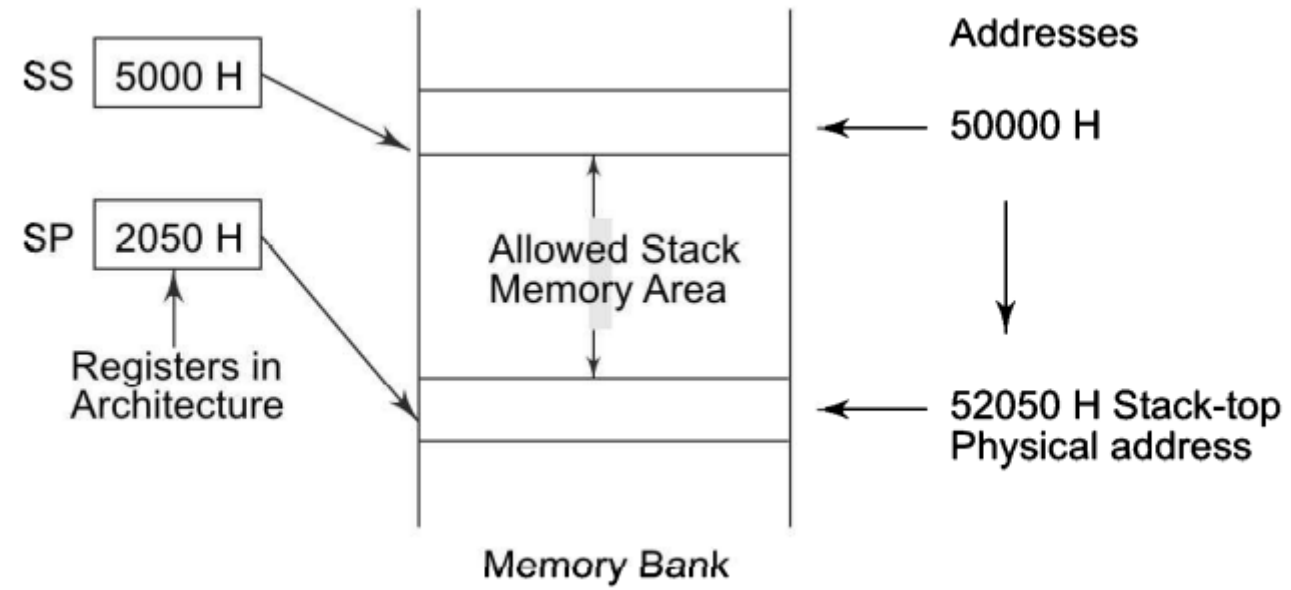
10H * SS \Rightarrow 0101 0000 0000 0000 0000

+

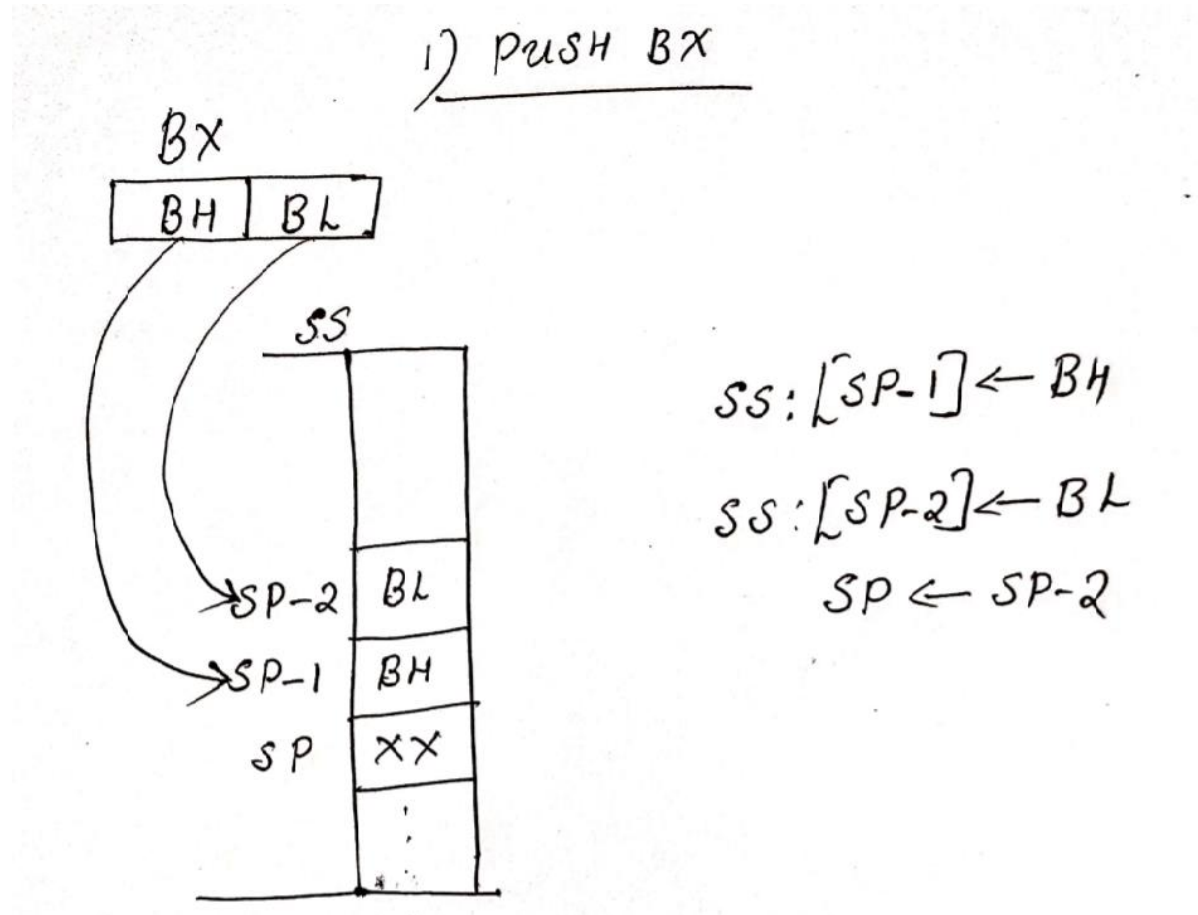
SP \Rightarrow 0010 0000 0101 0000

Stack-top 0101 0010 0000 0101 0000

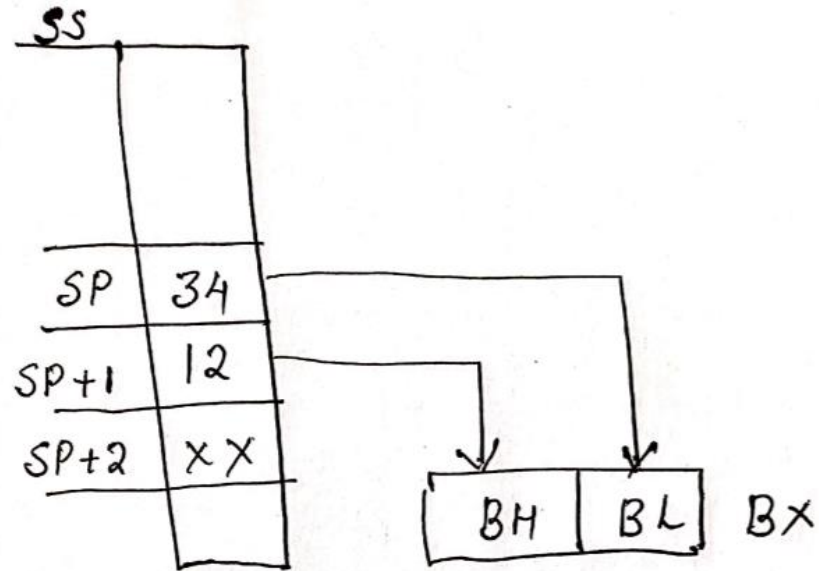
address 5 2 0 5 0



PUSH BX



POP BX



$BL \leftarrow SS:[SP]$

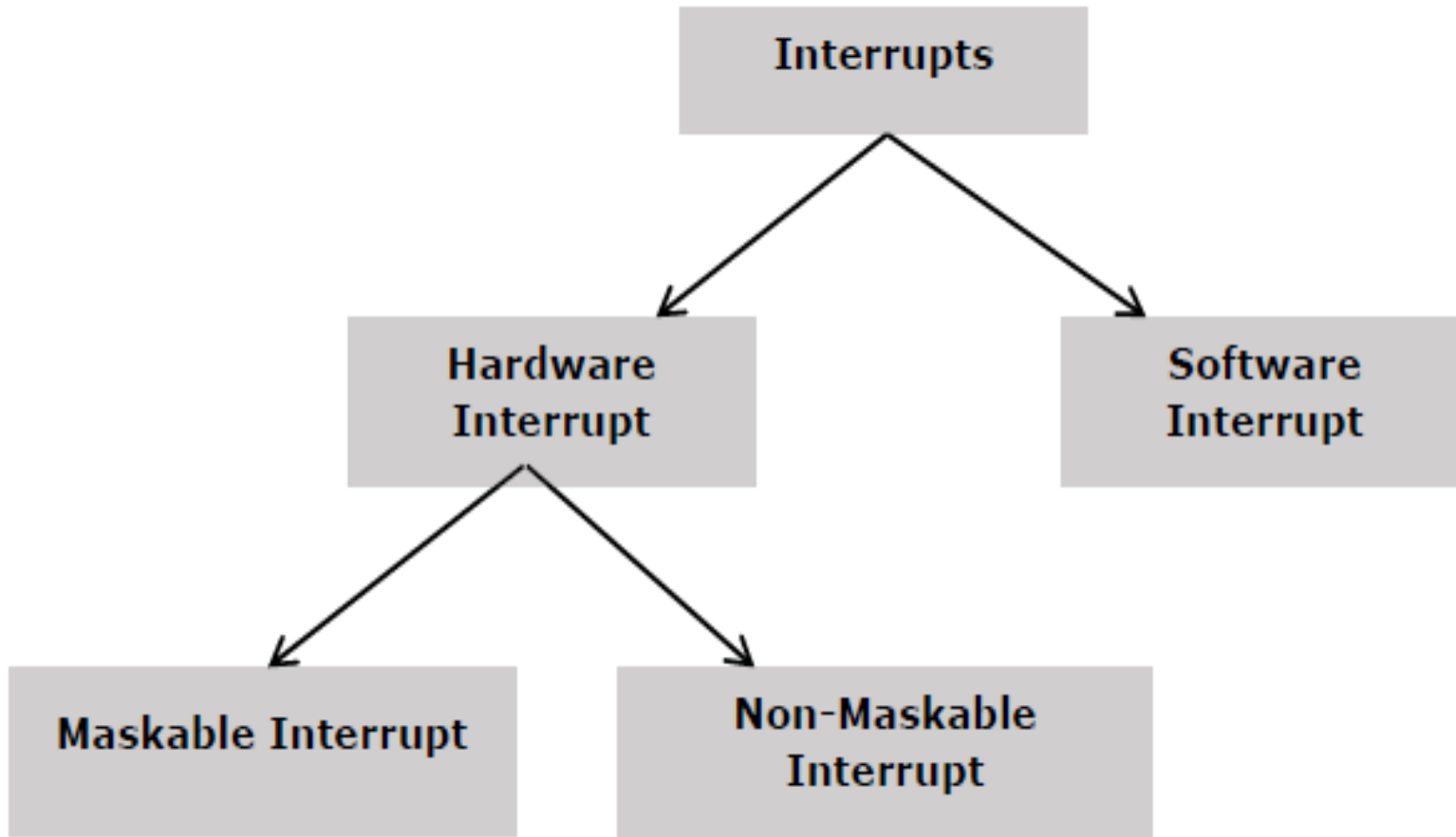
$BH \leftarrow SS:[SP+1]$

$SP \leftarrow SP+2$

Interrupts

- **Interrupt** is the method of creating a temporary halt during program execution and allows peripheral devices to access the microprocessor.
- The microprocessor responds to that interrupt with an **ISR** (Interrupt Service Routine)

Types of interrupts in 8086 microprocessor



Hardware Interrupts

- Caused by any peripheral device by sending a signal through a specified pin
- The 8086 has two hardware interrupt pins, i.e. NMI and INTR
- NMI is a non-maskable interrupt and INTR is a maskable interrupt having lower priority.
- INTA is called interrupt acknowledge.

Hardware Interrupts Contd..

NMI

- Non-maskable interrupt pin (NMI) having higher priority than the maskable interrupt request pin (INTR)

When this interrupt is activated, these actions take place –

- Completes the current instruction that is in progress.
- Pushes the Flag register values on to the stack.
- Pushes the CS (code segment) value and IP (instruction pointer) value of the return address on to the stack.
- IP is loaded from the contents of the word location 00008H.
- CS is loaded from the contents of the next word location 0000AH.
- Interrupt flag and trap flag are reset to 0.

INTR

The INTR is a maskable interrupt because the microprocessor will be interrupted only if interrupts are enabled using set interrupt flag instruction(STI)

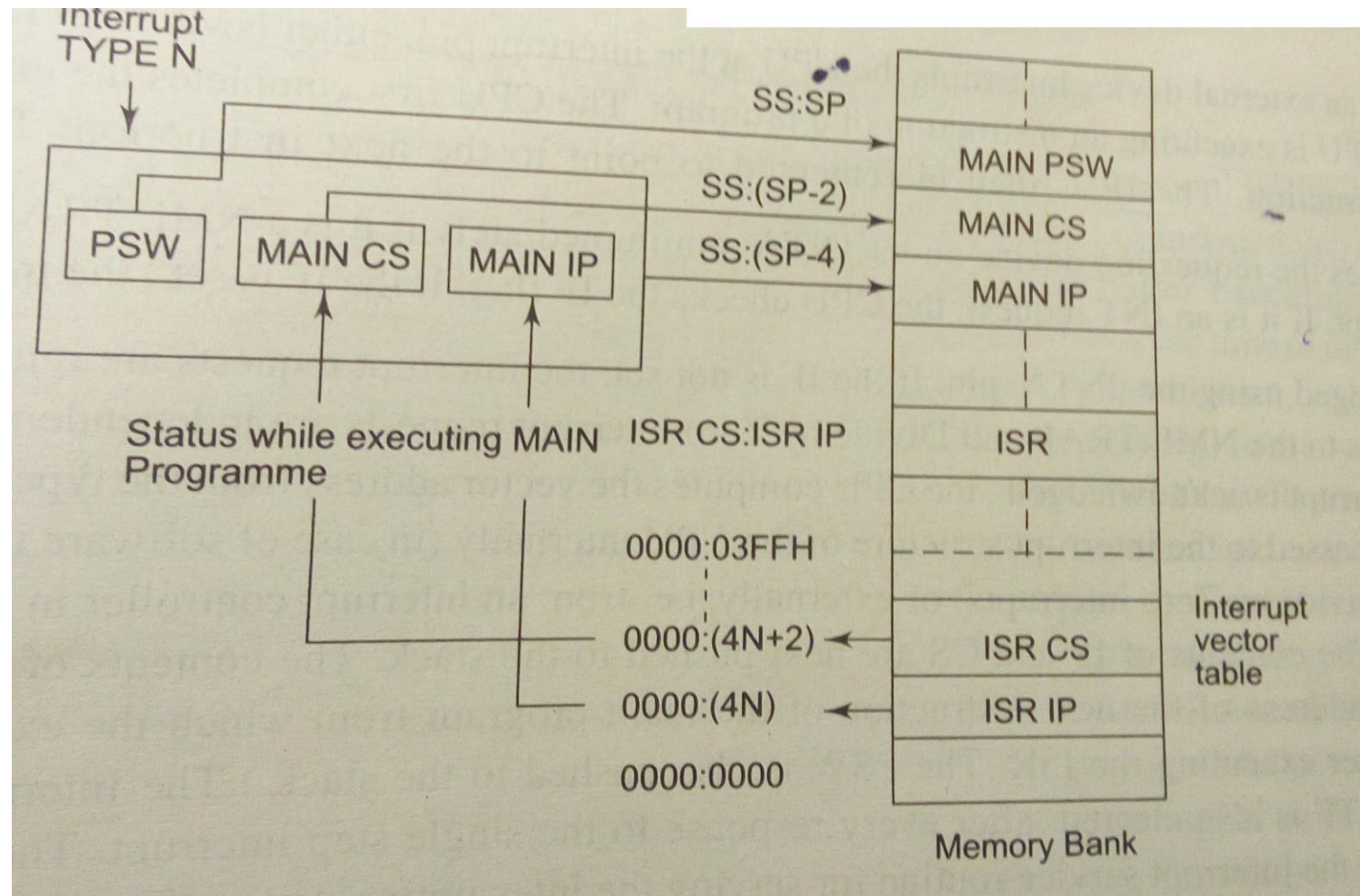
These actions are taken by the microprocessor –

- First completes the current instruction.
- Activates INTA output and receives the interrupt type, say X.
- Flag register value, CS value of the return address and IP value of the return address are pushed on to the stack.
- IP value is loaded from the contents of word location $X \times 4$
- CS is loaded from the contents of the next word location 0000H
- Interrupt flag and trap flag is reset to 0

Software Interrupts

- Some instructions are inserted at the desired position into the program to create interrupts
- There are 256 interrupt types under this group.

Interrupt Response Sequence



Structure of Interrupt Vector Table

Interrupt Type	Content (16-bit)	Address	Comments
Type 0	ISR IP	0000:0000	Reserved for divide by Zero interrupt
	ISR CS	0000:0002	
Type 1	ISR IP	0000:0004	Reserved for single step interrupt
	ISR CS	0000:0006	
Type 2	ISR IP	0000:0008	Reserved for NMI
	ISR CS	0000:000A	
Type 3	ISR IP	0000:000C	Reserved for INT single byte instruction
	ISR CS	0000:000E	
Type 4	ISR IP	0000:0010	Reserved for INTO instruction
	ISR CS	0000:0012	
Type N		0000:0014	Reserved for two byte instruction INT TYPE
		0000:0016	
	ISR IP	0000:004N	
	ISR CS	0000:(004N+2)	
		0000:03FC	
	ISR IP	0000:03FE	
	ISR CS	0000:03FF	
Type FFH			

ISR : Interrupt Service Routine

Interrupts of 8086

- 8086 has 256 types of interrupts
- Type-0 to Type-4 are dedicated for specific functions by INTEL and they are called INTEL predefined interrupts
- Type-5 to Type-31 are reserved by INTEL
- Type-32 to Type-255 are available for the user as hardware or software interrupts

INTEL predefined or dedicated interrupts

1) Divide by zero interrupt(Type-0)

- Implemented as a part of divide instruction
- 8086 automatically do type-0 interrupt
- Non Maskable
- ISR should be stored in m/m
- Address of ISR is stored in IVT

2) Single step(Type-1) interrupt

- Generate when $TF=1$
- User can write ISR for Type-1 interrupt
- After execution of each instruction, user can halt processor and return control to user
- This feature will be useful to debug a program

- **3) NMI (Type-2) interrupt**

Automatically generate when it receives ***low-to-high*** transition on its NMI pin

Used to save program data or status in case of system ac power failure

4) Break point(Type-3) interrupt

- Implement a break point function
- Execute a program partly or up to a desired point and then return control to user
- Useful to debug a program

5) Overflow (Type-4) interrupt

- This interrupt is initiated by the instruction INTO
- Overflow flag will be set

Interrupt priority(high to low)

- 1)Divide error, INT n, INTO
- 2)NMI
- 3)INTR
- 4)SINGLE STEP

Vectored and Non Vectored Interrupts

- If a program control automatically branches to a specified address when an interrupt signal is accepted by the processor. Such an interrupt is called **vectored interrupt**
- In **non vectored interrupt** the interrupting device should supply the address of the ISR to be executed in response to the interrupt
- All the 8086 interrupt are vectored

How 8086 find out the address of an ISR

- Total 1024 bytes are reserved for interrupt vector table
- Each interrupt requires 4 bytes
- IVT contains IP and CS of all interrupts stored sequentially from 0000:0000 to 0000:03FFH
- The interrupt type N is multiplied by 4 and hexadecimal multiplication obtained gives the offset address in the zero th code seg at which the IP and Cs addresses of ISR are stored.

- Question

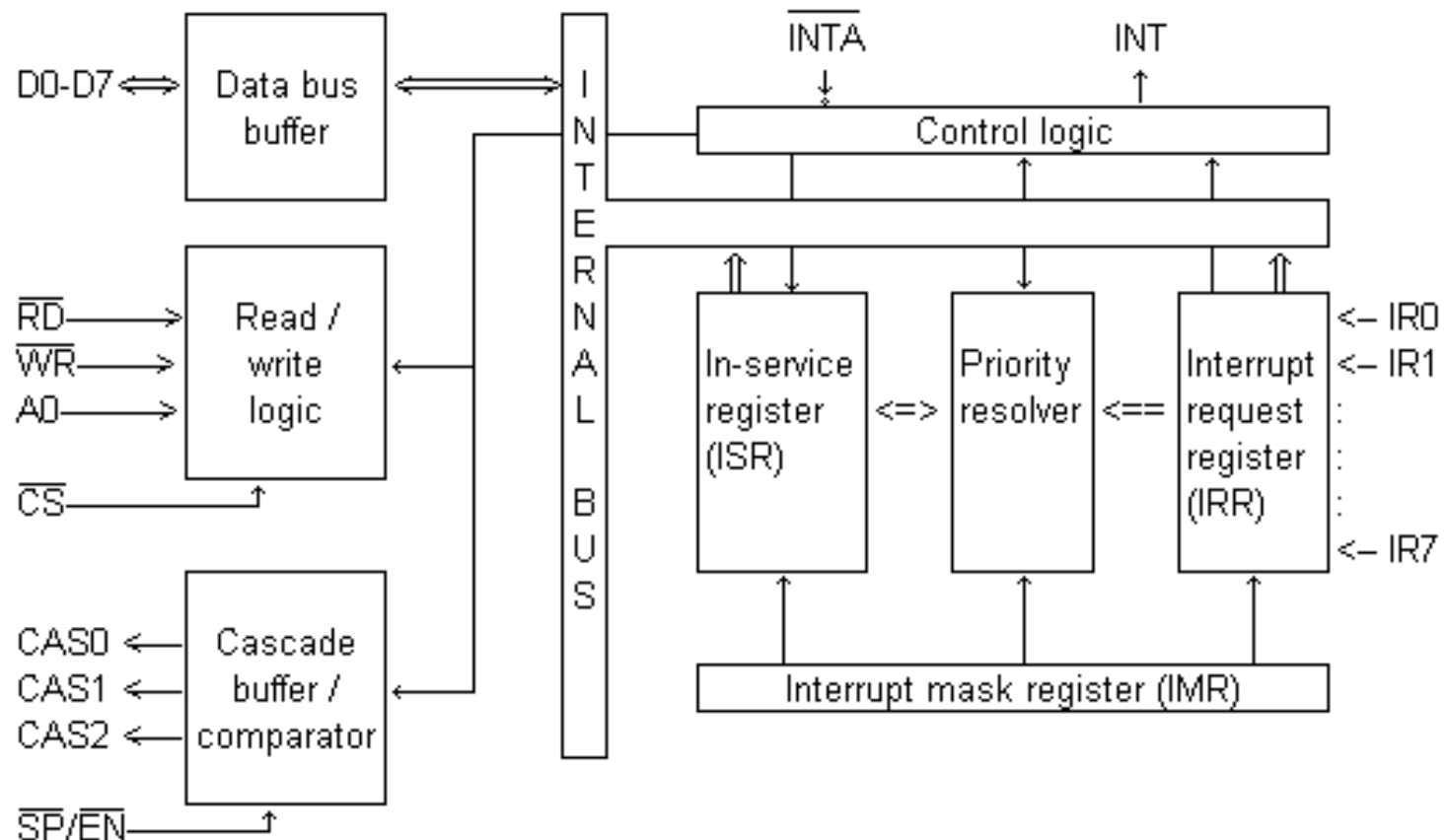
- 1) Find out the address of the ISR(IP and CS) in an 8086 MP, if an interrupt occurs in the program like INT 21H.

Module 3

PROGRAMMABLE INTERRUPT CONTROLLER (PIC) 8259A

PROGRAMMABLE INTERRUPT CONTROLLER (PIC) 8259A

8259 internal block diagram



Interrupt Request Register (IRR): The interrupts at IRQ input lines are handled by Interrupt Request Register internally. IRR stores all the interrupt request in it in order to serve them one by one on the priority basis.

In-Service Register (ISR): This stores all the interrupt requests those are being served, i.e. ISR keeps track of the requests being served

Priority Resolver : This unit determines the priorities of the interrupt requests appearing simultaneously. The highest priority is selected and stored into the corresponding bit of ISR during INTA pulse. The IR0 has the highest priority while the IR7 has the lowest one, normally in fixed priority mode. The priorities however may be altered by programming the 8259A in rotating priority mode.

Interrupt Mask Register (IMR) : This register stores the bits required to mask the interrupt inputs. IMR operates on IRR at the direction of the Priority Resolver.

- **Interrupt Control Logic** : This block manages the interrupt and interrupt acknowledge signals to be sent to the CPU for serving one of the eight interrupt requests. This also accepts the interrupt acknowledge (INTA) signal from CPU that causes the 8259A to release vector address on to the data bus.
- **Data Bus Buffer** : This tristate bidirectional buffer interfaces internal 8259A bus to the microprocessor system data bus. Control words, status and vector information pass through data buffer during read or write operations.

Read/Write Control Logic : This circuit accepts and decodes commands from the CPU. This block also allows the status of the 8259A to be transferred on to the data bus.

Cascade Buffer/Comparator : This block stores and compares the ID's of all the 8259A used in system. The three I/O pins CAS0-CAS2 are outputs when the 8259A is used as a master. The same pins act as inputs when the 8259A is in slave mode. The 8259A in master mode sends the ID of the interrupting slave device on these lines. The slave thus selected, will send its preprogrammed vector address on the data bus during the next INTA pulse.

Interrupt Sequence in an 8086 system

One or more IR lines are raised high that set corresponding IRR bits

2. 8259A resolves priority and sends an INT signal to CPU
3. The CPU acknowledge with INTA pulse.
4. Upon receiving an INTA signal from the CPU, the highest priority ISR bit is set and the corresponding IRR bit is reset. The 8259A does not drive data during this period
5. The 8086 will initiate a second INTA pulse. During this period 8259A releases an 8-bit pointer on to a data bus from where it is read by the CPU
6. This completes the interrupt cycle. The ISR bit is reset at the end of the second INTA pulse if automatic end of interrupt (AEIOI) mode is programmed. Otherwise ISR bit remains set until an appropriate EOI command is issued at the end of interrupt subroutine

Programming of 8259

- 8259A has two types of command words, initialization(ICWs) and operational control word(OCWs)
 - i. Initialization command word:
 - a. ICW1 for chip (8259A) control
 - b. ICW2 for type
 - c. ICW3 for status control
 - d.ICW4 for mode control

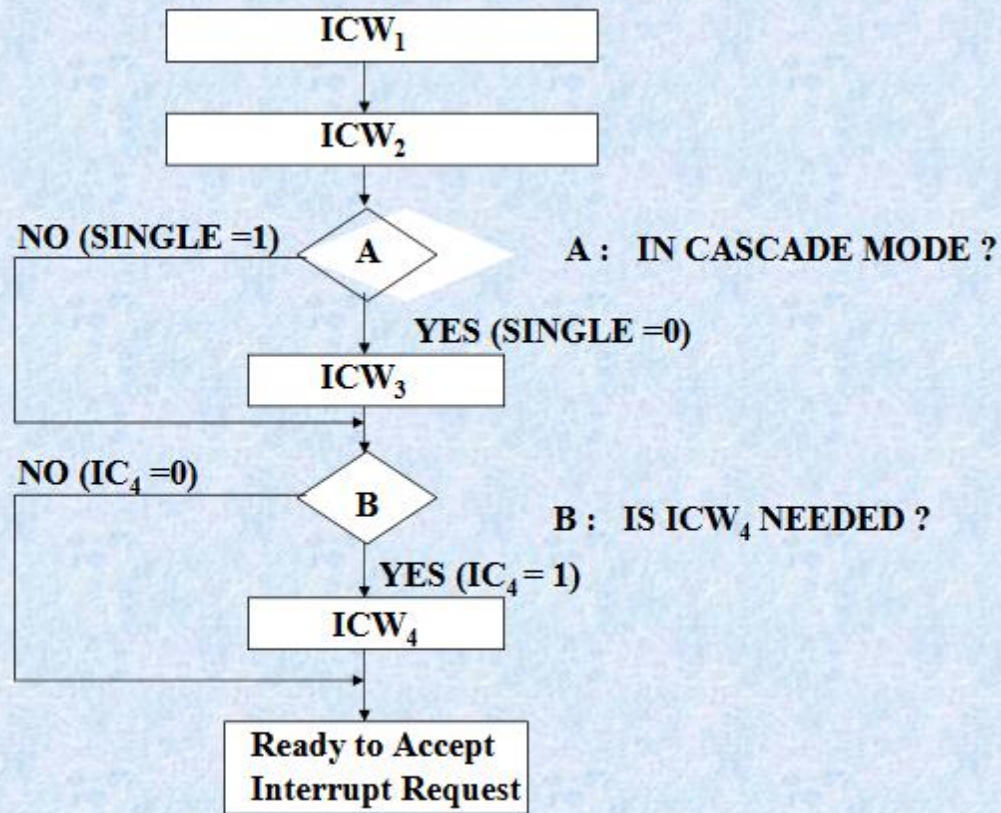


Fig 3: Initialisation Sequence of 8259A

Initialization command word

ICW1

- Control bits for edge/level triggered mode , single/cascade mode , call address interval and whether ICW4 is required or not needed

ICW2

- Stored details regarding interrupt vector addresses

Initialization command word

ICW3

- Used in cascaded mode

ICW4

- Used for mode control

ICW1

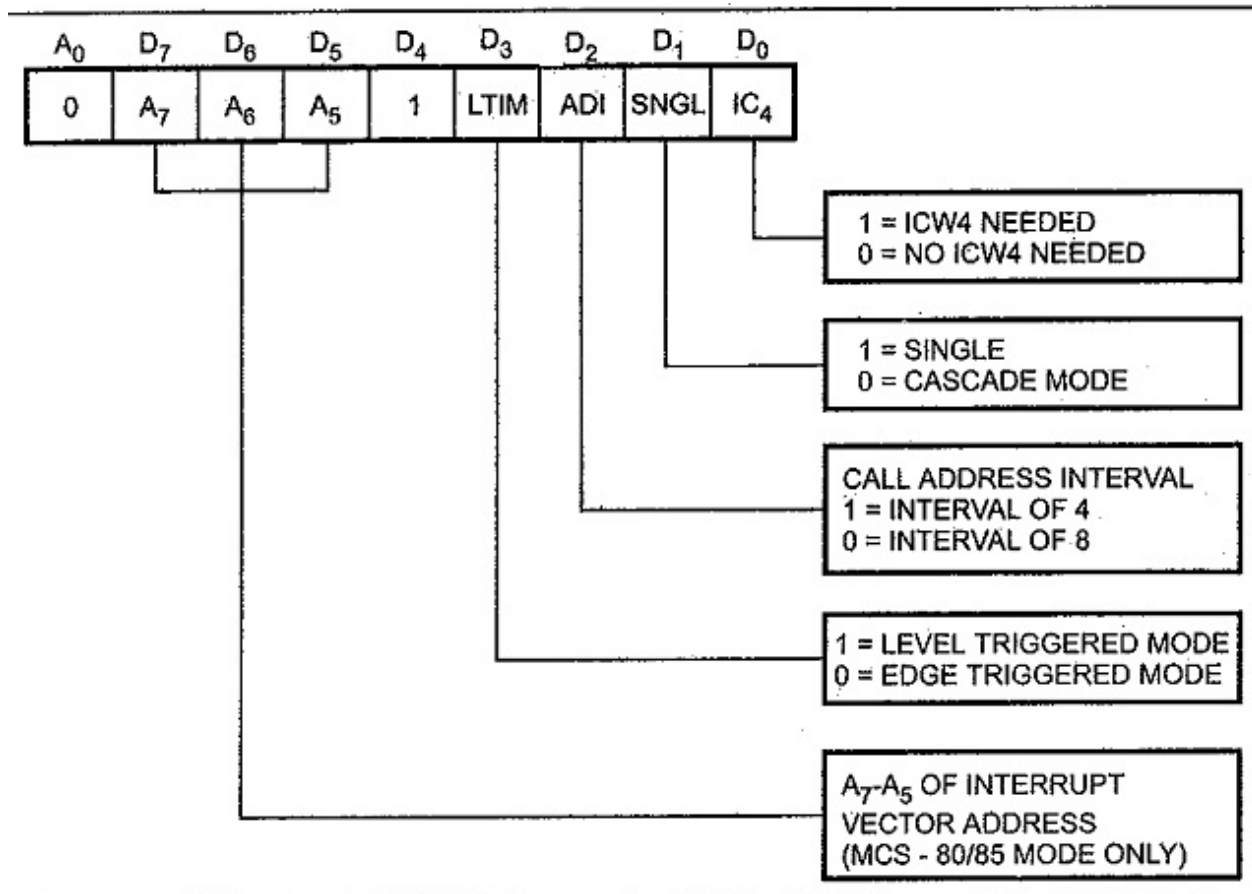


Fig. 14.76 Initialization command word 1 (ICW1)

ICW2, ICW3(master)

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
A ₁₅ /T ₂	A ₁₄ /T ₆	A ₁₃ /T ₅	A ₁₂ /T ₄	A ₁₁ /T ₃	A ₁₀	A ₉	A ₈

a) The format of ICW3 to be issued to master 8259A is,

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
S ₇	S ₆	S ₅	S ₄	S ₃	S ₂	S ₁	S ₀

ICW3(Slave)

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	Slave Device
0	0	0	0	0	ID ₂	ID ₁	ID ₀	
					0	0	0	0
					0	0	1	1
					0	1	0	2
					0	1	1	3
					1	0	0	4
					1	0	1	5
					1	1	0	6
					1	1	1	7

ICW4

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
0	0	0	SFNM	BUF	MS	AEOI	μPM

Operation Command words

- Modes of operations can be selected by programming, i.e. writing three internal registers called as operation command Words
- In the three operation command words OCW 1, OCW2 and OCW3 every bit corresponds to some operational feature of the mode selected, except for a few bits those are either 1 or 0.

OCW1

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
M_7	M_6	M_5	M_4	M_3	M_2	M_1	M_0

OCW2

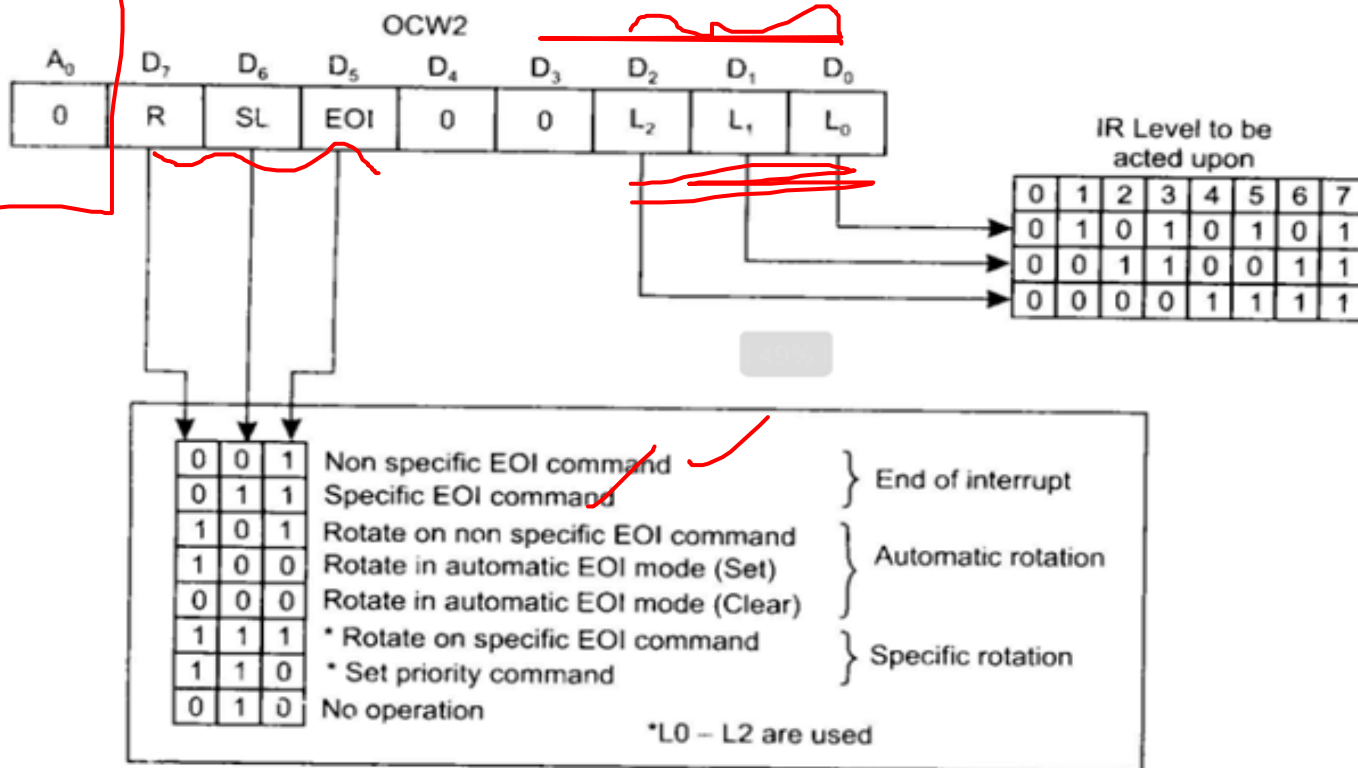


Fig. 9e.13: Format of operation command word 2
(Source: Intel Corporation)

OCW3

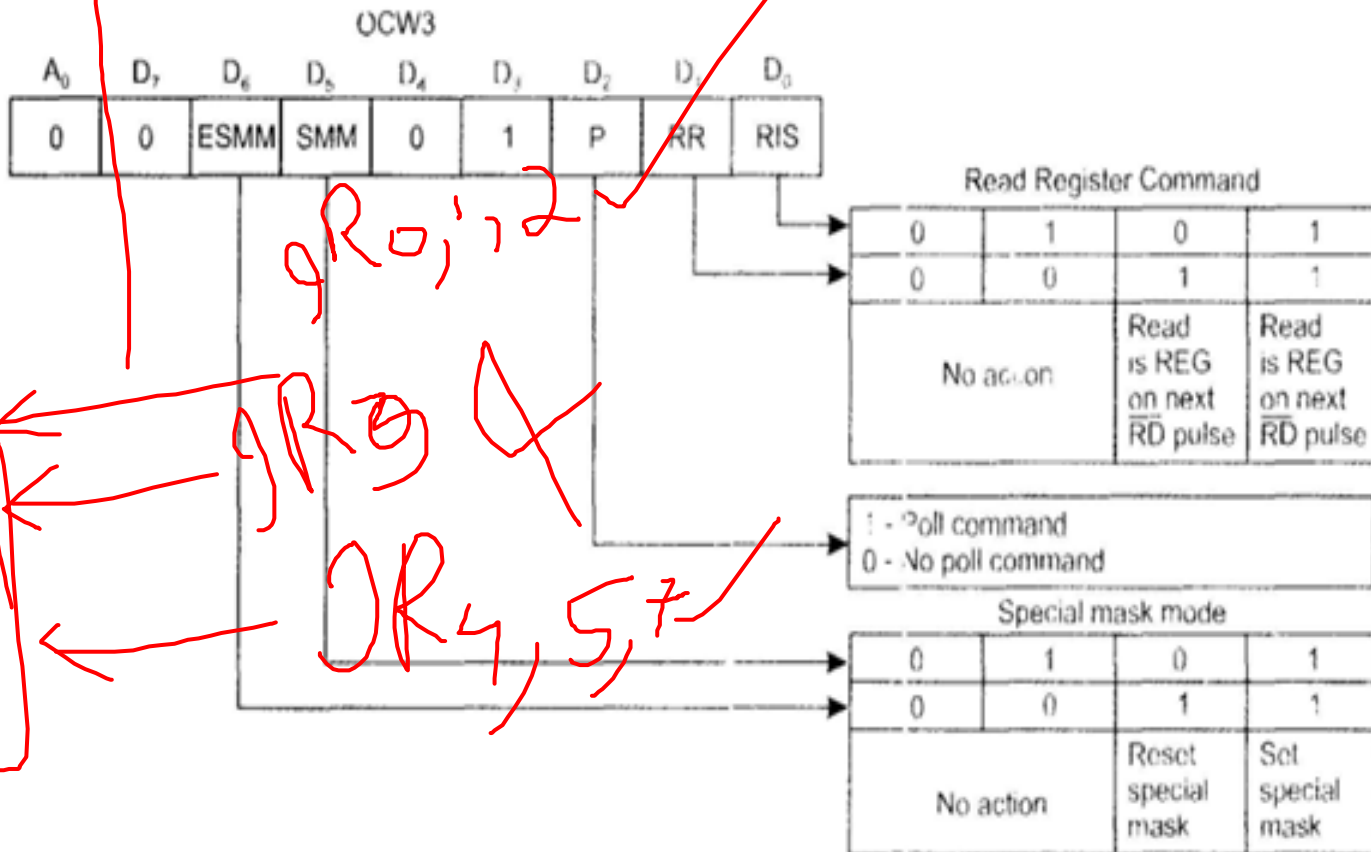
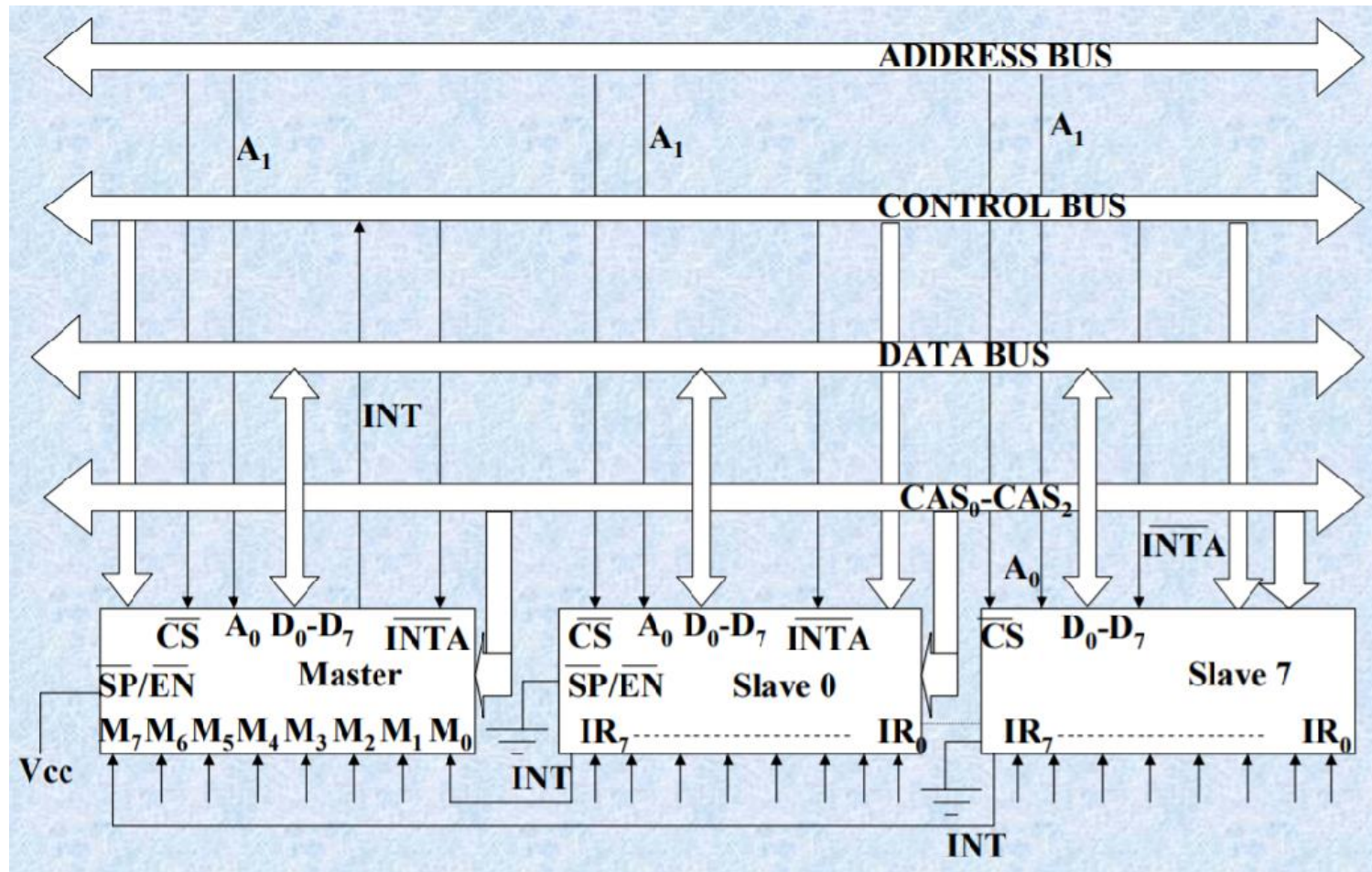
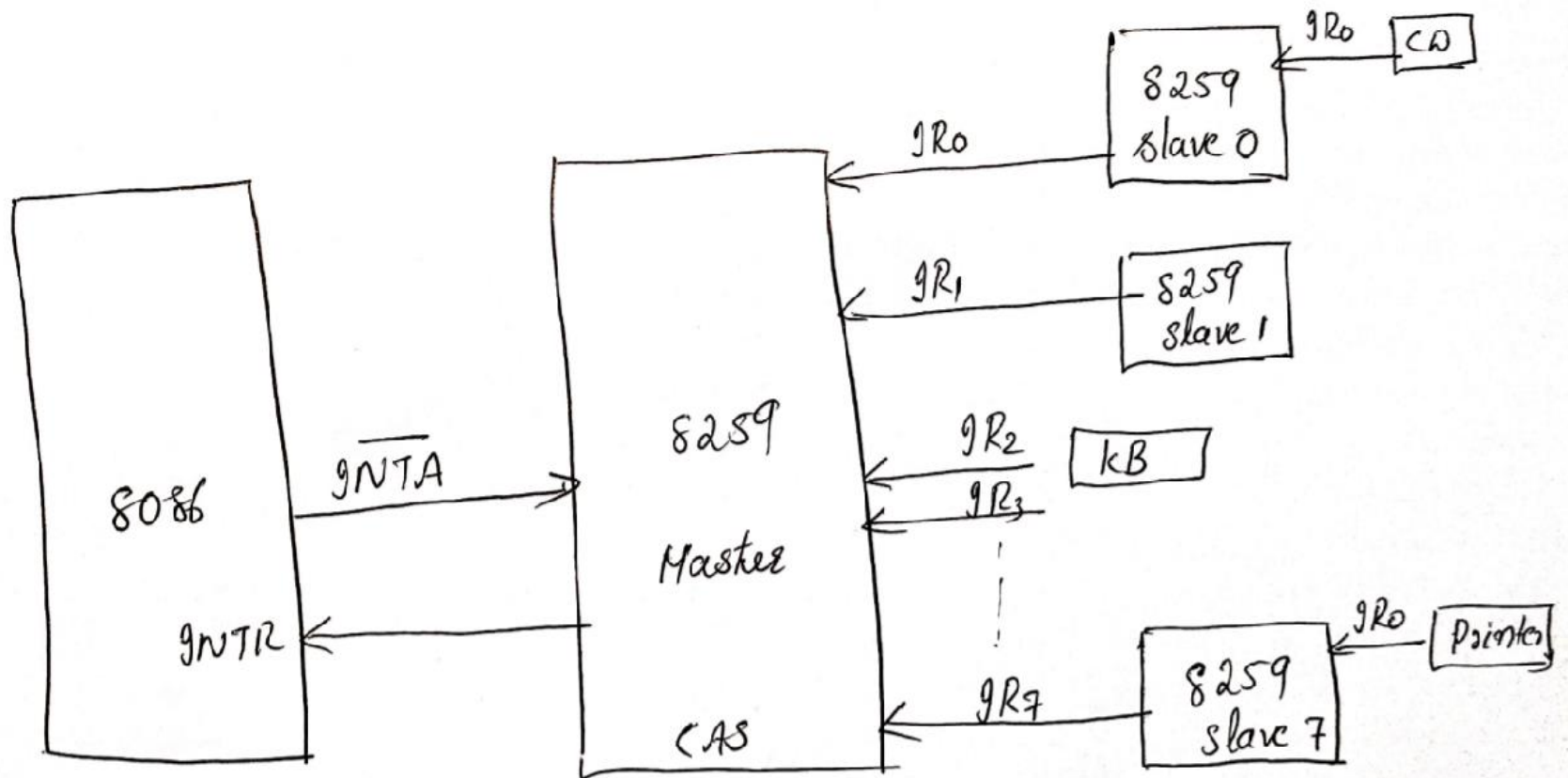


Fig. 9e.14: Format of operation command word 3
(Source: Intel Corporation)

8259 in cascaded mode





Operating Modes of 8259

- **Fully nested mode** : default priority mode, IR0 – highest, IR7-lowest priority , fixed, single
- **Specially Fully nested mode** : default, IR0 – highest, IR7-lowest priority , fixed, cascaded
- **Rotating Priority Modes-2 types**
 - Automatic rotation mode : Applicable where all interrupting devices are of equal priority . After servicing a device , it gets lowest priority . All other priorities rotates subsequently
 - Specific rotation mode: Here user can assign lower priority to any IR line and thus fix all other priorities

Operating Modes of 8259 Contd...


- **Special Mask mode(SMM)**

It inhibits further interrupts at that level and enables interrupt from other levels

- **Poll mode:**

Here INT line of 8259 is not used. In return 8259 gives poll command. Poll command indicates highest priority interrupt

Poll Word

I	x	x	x	x	W ₂	W ₁	W ₀	
1 = Valid Interrupt 0 = No valid interrupt						0	0	0
						0	0	1
						0	1	0
						0	1	1
Level No of the highest priority interrupt to be serviced						1	0	0
						1	0	1
						1	1	0
						1	1	1

- End of Interrupt modes (EOI):

Normal EOI mode

- Non Specific EOI mode: Prgmer does nt specify the bit to be cleared. 8259 automatically clears the bit with highest priority from InSR
- Specific EOI mode: Prgmer specify the bit to be cleared from InSR

Auto EOI mode(AEOI): EOI is not needed .8259 itself clear he corresponding bit from InSR

ASSUME CS : CODE, DS : DATA

DATA SEGMENT

FILENAME DB "RESULT", "\$"

MESSAGE DB "FILE WASN'T CREATED SUCCESSFULLY", 0AH, 0DH, "\$"

DATA ENDS

```

CODE SEGMENT
START: MOV AX, CODE
      MOV DS, AX
      MOV DX, OFFSET ISROA
      MOV AX, 250AH
      INT 21H
      MOV DX, OFFSET FILENAME
      MOV AX, DATA
      MOV DS, AX
      MOV CX, 00H
      MOV AH, 3CH
      INT 21H
      JNC FURTHER
      MOV DX, OFFSET MESSAGE
      MOV AH, 09H
      INT 21H
      JMP STOP
FURTHER : INT 0AH
STOP :    MOV AH, 4CH
          INT 21H

```

```

ISROA PROC NEAR
      MOV BX, AX
      MOV CX, 500H
      MOV DX, 1000H
      MOV AX, 1000H
      MOV DS, AX

```

MOV AH, 40 H

INT 21 H

IRET

ISR0A ENDP

CODE ENDS

END START

MEMORY INTERFACING

Interfacing Memory

Static Memory Interfacing

The semiconductor memories are organised as two dimensional arrays of memory locations.

- For example 4K * 8 or 4K byte memory contains 4096 locations, where each locations contains 8-bit data and only one of the 4096 locations can be selected at a time. Once a location is selected all the bits in it are accessible using a group of conductors called Data bus.
- For addressing the 4K bytes of memory, 12 address lines are required.
- In general to address a memory location out of N memory locations, will require at least n bits of address, i.e. n address lines where $n = \log_2 N$.
- Thus if the microprocessor has n address lines, then it is able to address at the most N locations of memory, where $2^n = N$.

- If out of N locations only P memory locations are to be interfaced, then the least significant p address lines out of the available n lines can be directly connected from the microprocessor to the memory chip while the remaining $(n-p)$ higher order address lines may be used for address decoding as inputs to the chip selection logic.
- The memory address depends upon the hardware circuit used for decoding the chip select (CS). The output of the decoding circuit is connected with the CS pin of the memory chip.

The general procedure of static memory interfacing with 8086

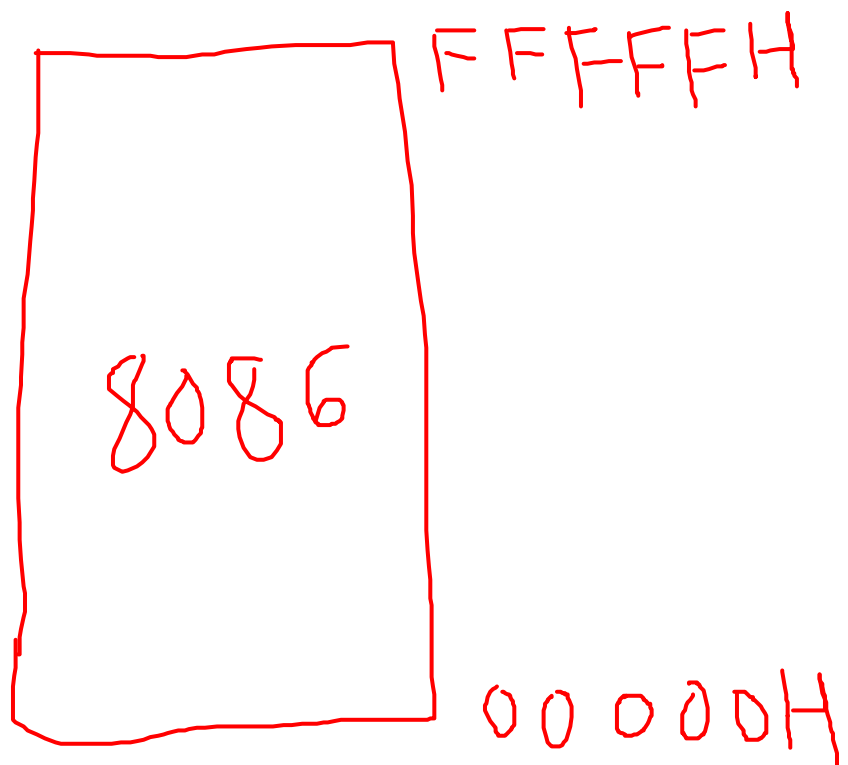
1. Arrange the available memory chips so as to obtain 16-bit data bus width.

The upper 8-bit bank is called 'odd address memory bank' and the lower 8-bit bank is called 'even address memory bank'.

2. Connect available memory address lines of memory chips with those of the microprocessor and also connect the memory RD and WR inputs to the corresponding processor control signals. Connect the 16-bit data bus of the memory bank with that of the microprocessor 8086.

3. The remaining address lines of the microprocessor, BHE and A₀ are used for decoding the required chip select signals for the odd and even memory banks. The CS of memory is derived from the output of the decoding circuit.

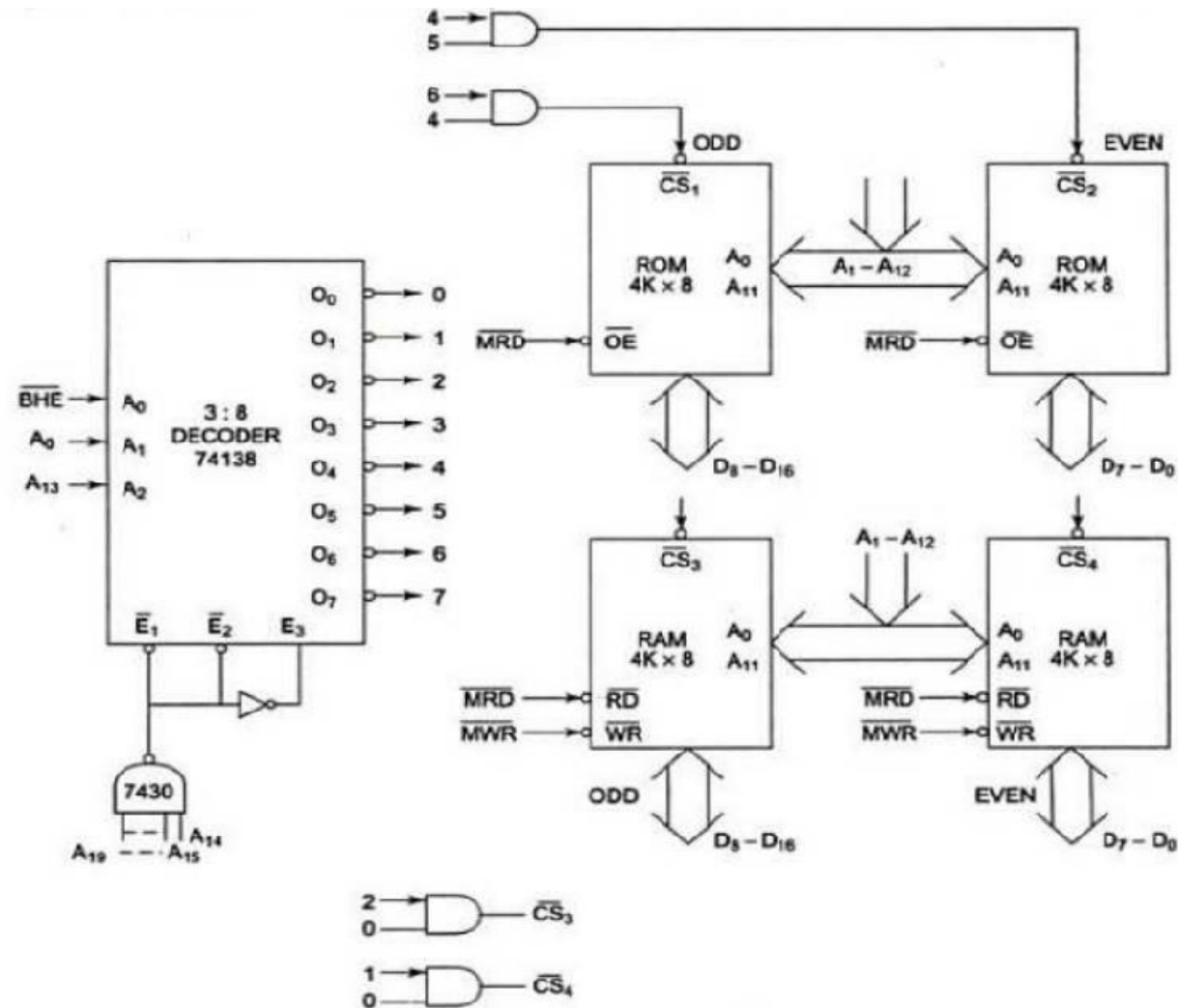
4. As a good and efficient interfacing practice, the address map of the system should be continuous as far as possible



Interface two 4Kx8 EPROMS and two 4Kx8 RAM chips with 8086. select suitable maps.

[illegible]

Interfacing diagram for the memory system



- The memory system in this example contains in total four 4Kx8 memory chip.
- The two 4Kx8 chips of RAM and ROM are arranged in parallel to obtain 16-bit data bus width. A_0 is 0, i.e. the address is even and is in RAM, then the lower RAM chip is selected indicating 8-bit transfer at an even address.
- If A_0 is 1, i.e. the address is odd and is in RAM, the BHE goes low, the upper RAM chip is selected, further indicating that the 8-bit transfer is at an odd address.

The selection of chips takes place as shown in table.

Memory Chip Selection Table:

Decoder I/P --> Address/ \overline{BHE} -->	A2 A13	A1 A0	A0 \overline{BHE}	Selection/ Comment
Word transfer on D0 - D15	0	0	0	Even and odd address in RAM
Byte transfer on D7 - D0	0	0	1	Only even address in RAM
Byte transfer on D8 - D15	0	1	0	Only odd address in RAM
Word transfer on D0 - D15	1	0	0	Even and odd address in RAM
Byte transfer on D7 - D0	1	0	1	Only even address in RAM
Byte transfer on D8 - D15	1	1	0	Only odd address in ROM

2) Design an 8086 based system having 32K EPROM using 16 kb chips & 128K RAM using 32K chips

3) Design an interface between 8086 CPU and two chips of 16K×8 EPROM and two chips of 32K×8 RAM. Select the starting address of EPROM suitably. The RAM address must start at 00000 H.

2) Design an 8086 based system having 32K EPROM using 16 kb chips & 128K RAM using 32K chips. The RAM address must start at 00000H

Memory Calculations:

EPROM:

Required = 32 KB, Available = 16 KB

No. of chips = 2 chips.

Starting address of EPROM is calculated as:

FFFFFH – (Space required by total EPROM of 32 KB)

$$\begin{array}{r} \text{F F F F F H} \\ - \text{7 F F F H} \\ \hline \text{F 8 0 0 0 H} \end{array}$$

Size of a single EPROM chip = 16 KB

$$\begin{aligned} &= 16 \times 1\text{KB} = 2^4 \times 2^{10} \\ &= 2^{14} \\ &= \underline{14} \text{ address lines} \\ &= \underline{\text{(A14 ... A1)}} \end{aligned}$$

RAM:

Required = 128 KB, Available = 32 KB

No. of chips = 4 chips.

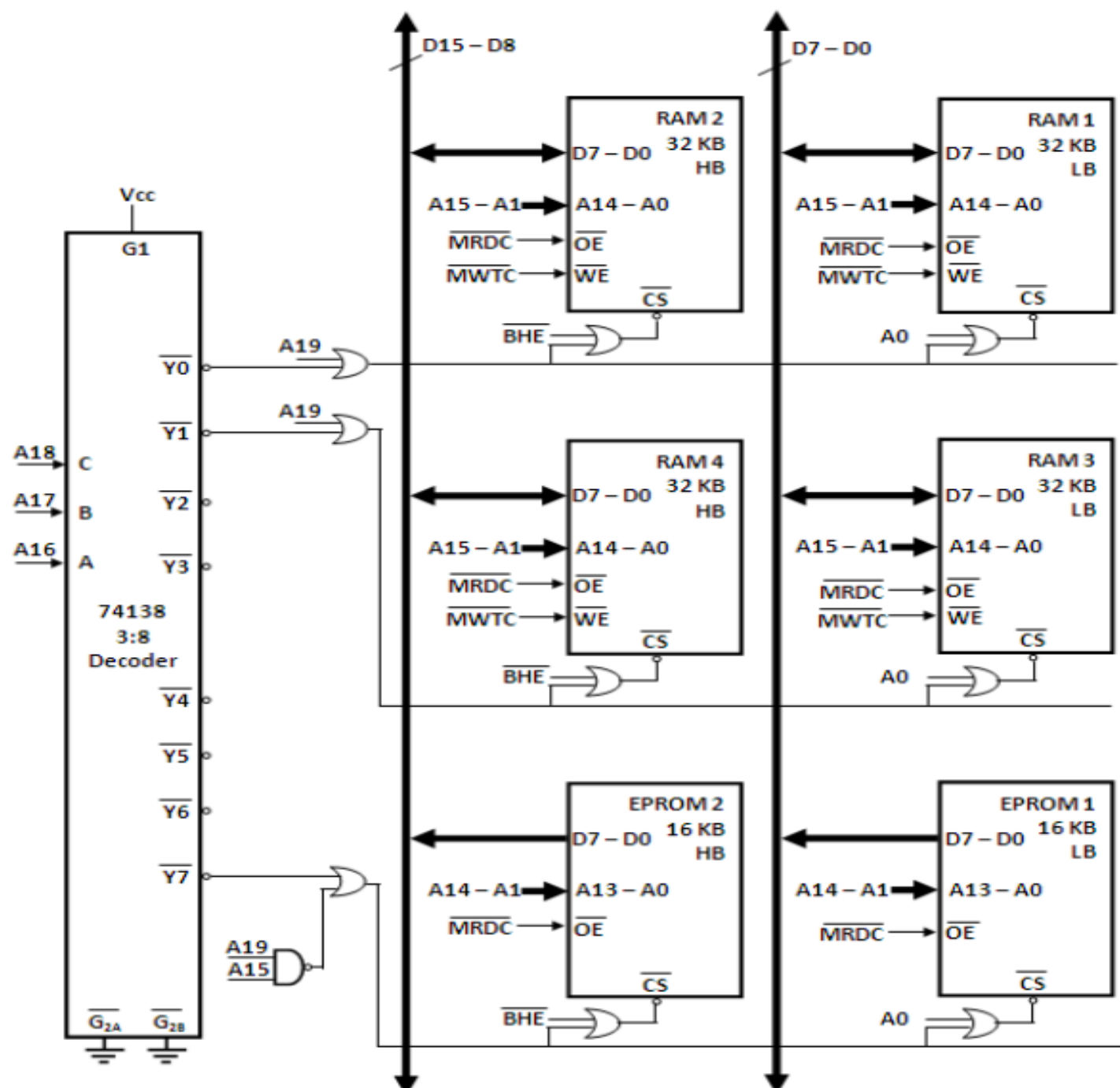
Starting address of RAM is: 00000H

Size of a single RAM chip = 32 KB

$$\begin{aligned} &= 32 \times 1 \text{ KB} = 2^5 \times 2^{10} \\ &= 2^{15} \\ &= \underline{15} \text{ address lines} \\ &= \underline{\text{(A15 ... A1)}} \end{aligned}$$

MEMORY MAP

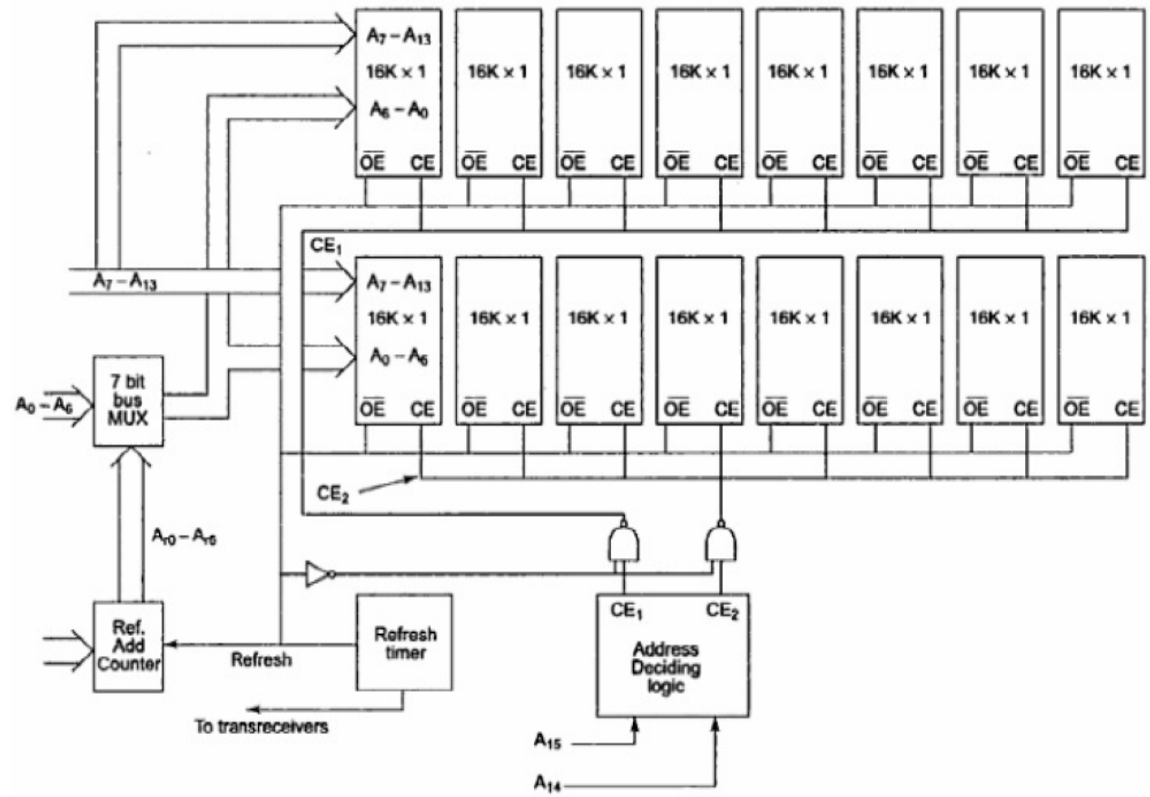
[illegible]



Dynamic RAM Interfacing

- The basic Dynamic RAM cell uses a capacitor to store the charge as a representation of data. This capacitor is manufactured as a diode that is reverse-biased so that the storage capacitance comes into the picture.
- This storage capacitance is utilized for storing the charge representation of data.
- The reverse-biased diode has a leakage current that tends to discharge the capacitor giving rise to the possibility of data loss.
- To avoid this possible data loss, the data stored in a dynamic RAM cell must be refreshed after a fixed time interval regularly.
- The process of refreshing the data in the RAM is known as refresh cycle.
- During this refresh period all other operations (accesses) related to the memory subsystem are suspended.

- The advantages of dynamic RAM. Like low power consumption, higher packaging density and low cost, most of the advanced computer systems are designed using dynamic RAMs.
- Also the refresh mechanism and the additional hardware required makes the interfacing hardware, in case of dynamic RAM, more complicated



- Generally dynamic RAM is available in units of several Kilobits to even Megabits of memory.
- This memory is arranged internally in a two dimensional matrix array so that it will have n rows and m columns.
- The diagram shown in figure explains the refreshing logic and 8086 interfacing with dynamic RAM.
- Each of the used chips 16K * 1-bit Dynamic RAM cell array.
- The system contains two 16 Kbytes Dynamic RAM units.
- All the address and the data lines are assumed to be available from an 8086 microprocessor system.
- The OE pin controls output data buffers of the memory chip.
- The CE pins are active high chip select of memory chips.
- The refresh cycle starts, if the refresh output of the refresh timer goes high. OE and CE tends to be high
- The high CE enables the memory chip for refreshing .

I/O Interfacing Techniques

- Input/output devices can be interfaced with microprocessor systems in two ways :
 - 1. I/O mapped I/O**
 - 2. Memory mapped I/O**
- **1. I/O mapped I/O :**
 - The devices are viewed as distinct I/O devices and are addressed accordingly
 - All the available address lines are not used for interfacing
 - The I/O mapped device requires IN and OUT instructions for accessing them
 - Requires less hardware for decoding as less no: of address lines used
 - Max of 64k input and 64K output devices or 32K input and 32K output devices can be interfaced
 - In addition to data and address buses to address device, IORD and IOWR signals are used for I/O mapped interfacing

- **. Memory mapped I/O**

- The devices are viewed as memory locations and are addressed likewise
- In this type of I/O interfacing, the 8086 uses 20 address lines to identify an I/O device.
- Can have 1M memory-mapped input and output devices
- MRDC and MRTC signals are used for interfacing in memory-mapped I/O scheme
- All the applicable data transfer instructions can be used to communicate with memory-mapped I/O devices
- Complex decoding hardware is required