Interrupt

Interrupt

An **interrupt** is a condition that causes the **microprocessor** to temporarily work on a different task, and then later return to its previous task. **Interrupts** can be internal or external.

Software Interrupt (INT)

Hardware Interrupt

-Non-maskable Interrupt (NMI)

-Interrupt Input (INTR)

Response to an Interrupt

i. PUSH FLAGS

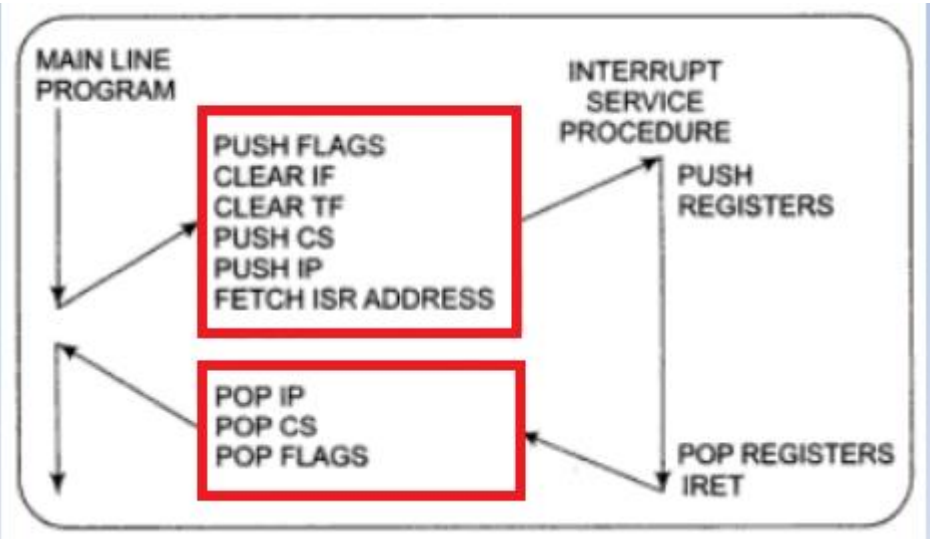
ii. Disables the 8086 INTR interrupt input by clearing the interrupt (IF) in the flag register

iii. It resets the trap flag (TF) in the flag register

iv. PUSH CS

v. PUSH IP

vi. It does an indirect far jump to the start of the procedure written to respond to the interrupt



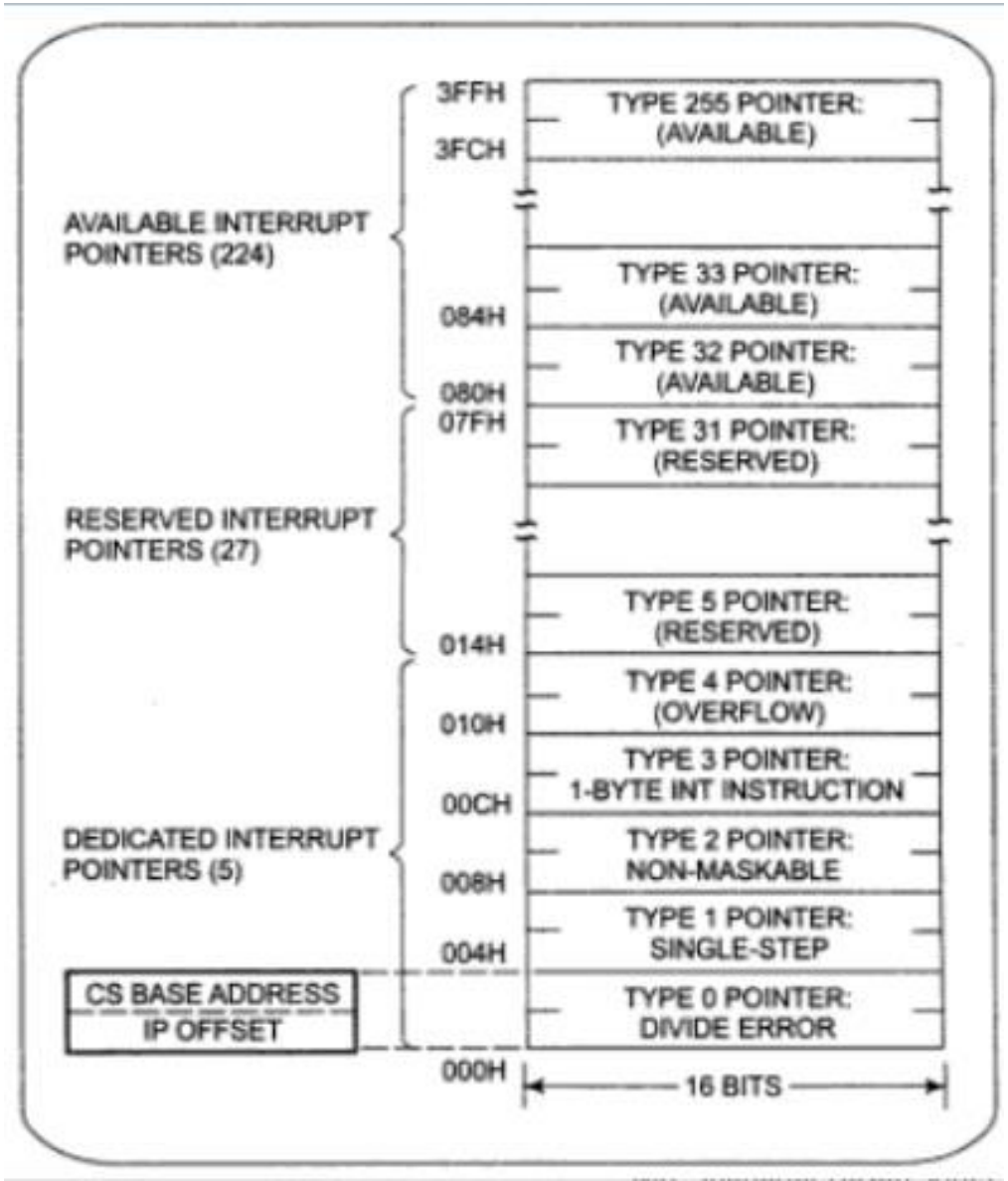
Return from an Interrupt (IRET)

- IRET restores the flags by popping in the flag register. This also reenables INTR.
- If INTR is still high, the program will again enter ISP.
- External hardware must be used to make sure that the signal is made low before you reenable INTR.
- You can reenable INTR within your ISP using the instruction : STI . Otherwise it will be active after reaching IRET of your ISP.

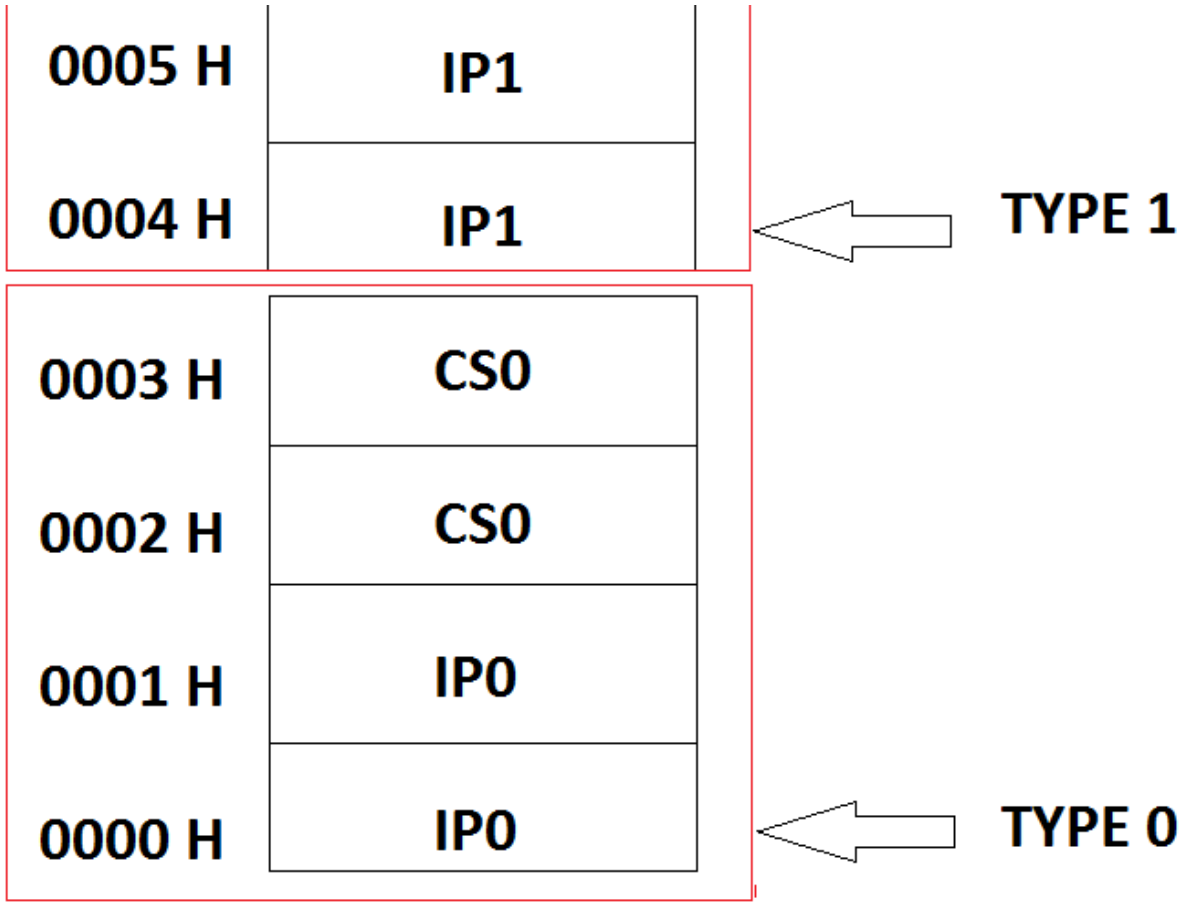
Interrupt Vector Table

- You need a dedicated memory space to store the staring CS-IP values of your ISPs.
- Each ISP requires a 2 Byte CS and a 2 Byte IP for its starting address.
- First 1 kB memory (00000H-003FFH) of 8086 is dedicated for storing CS and IP of ISPs.
- We can store 256 ISP in a single 8086
- Since the starting address of an ISP is called Interrupt Vector (or Interrupt Pointer) , this address table is called Interrupt Vector Table.

Interrupt Vector Table



Interrupt Vector Table



STI and CLI

- The 8086 INTR input allows some external signal to interrupt the execution of a program
- If the interrupt flag (IF) is cleared, then the INTR input is disabled (masked).

  CLI = Clear Interrupt -> IF = 0

  STI = Set Interrupt -> IF = 1
- These two instructions can be used anywhere in the program.
- When the 8086 is reset, IF is automatically cleared . This feature ensures that you will have the time necessary for initializations before any interrupt occurs.
- For INTR interrupts, the interrupt type is sent to the 8086 from an external device, Priority Interrupt Controller (8259A)

Interrupt Flag (IF)

- IF is automatically cleared when you enter an ISP and set when you leave. However you can still manually set IF with STI instruction anywhere within the ISP.
- It is important to clear IF upon entering ISP
  - Prevents a low priority interrupt while processing one with higher priority.
  - Does not allow same interrupt signal to incur multiple ISP operation.

Priority of Interrupts

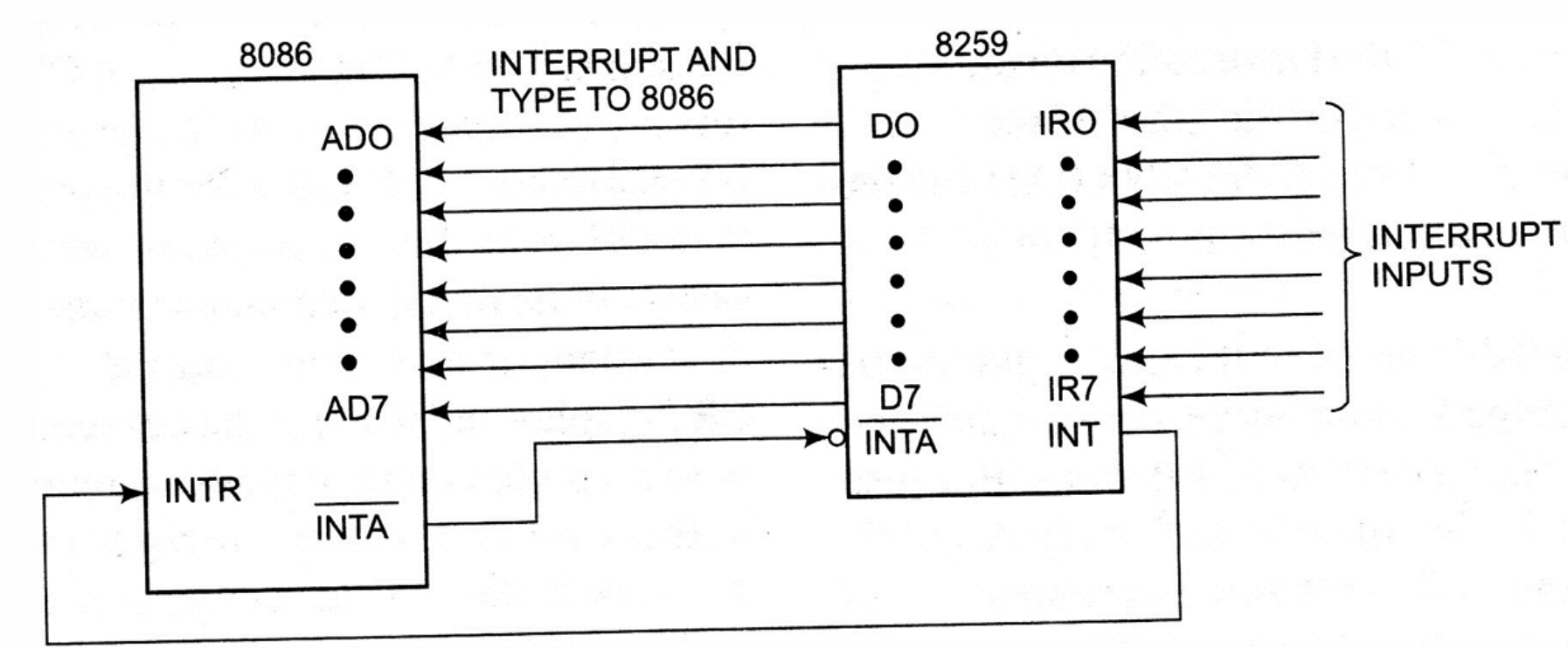
Interrupt	Priority
Divide Error, INT n, INTO	Highest
NMI	
INTR	
Single Step	Lowest

8086 has only 2 interrupt pins

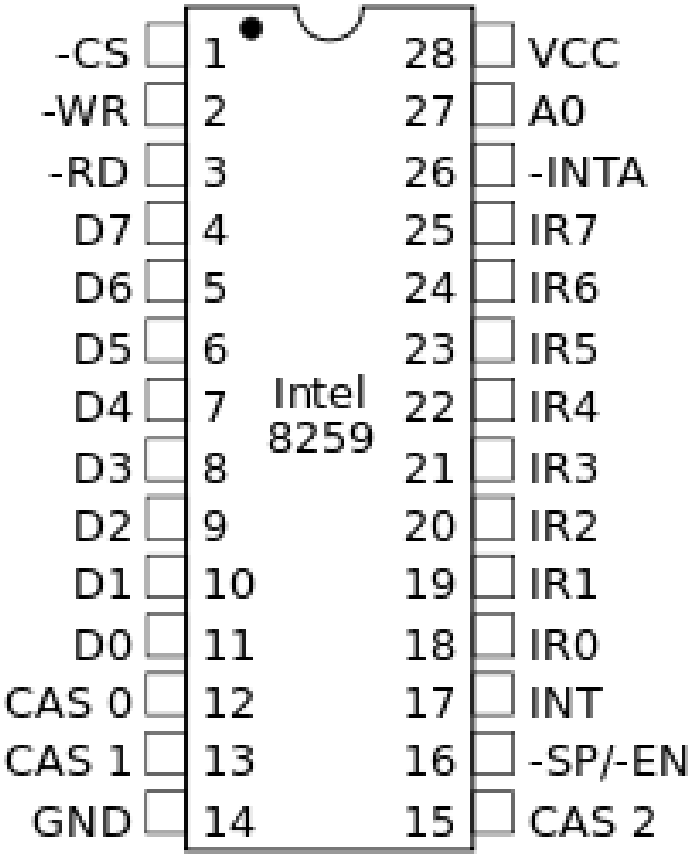
IF=1 means the processor is ready to take in any interrupt

the 8259 PIC is used to increase the interrupt pins of the processor

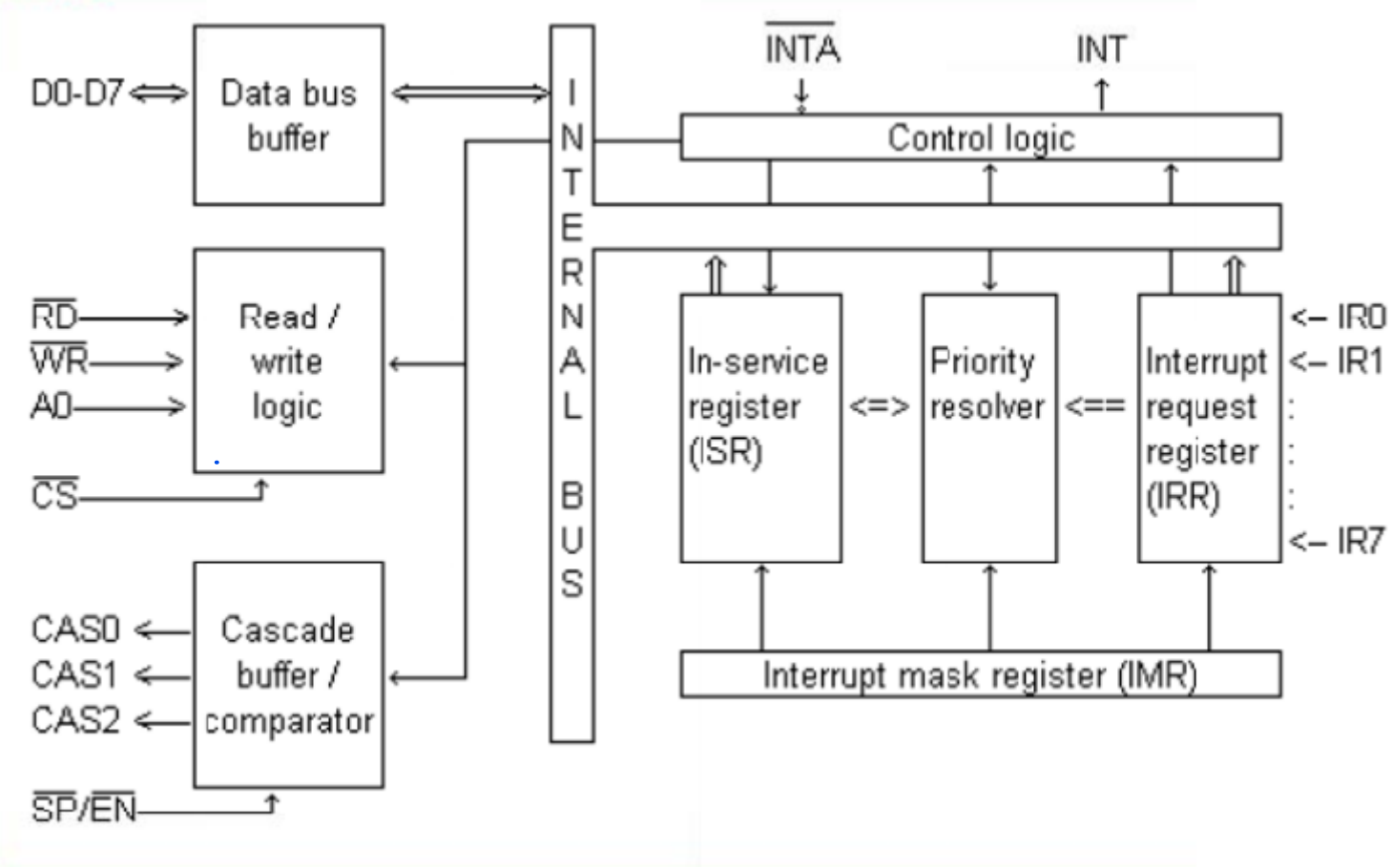
Priority Interrupt Controller



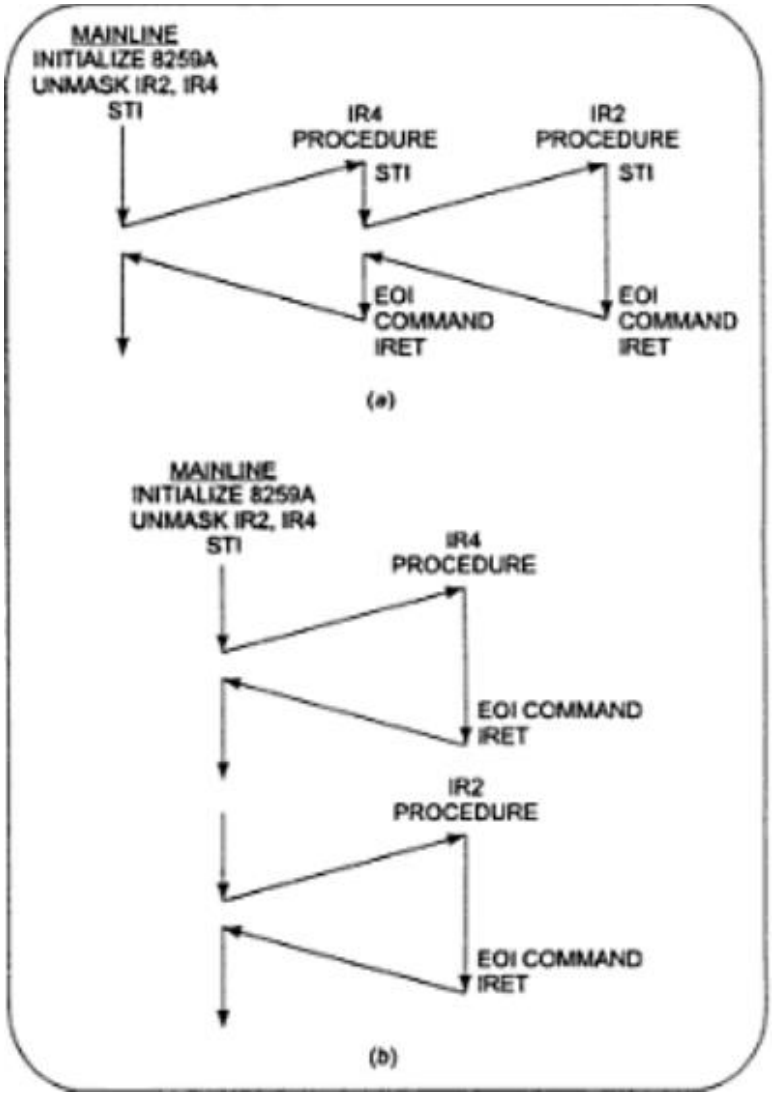
8259: Programmable Interrupt Controller



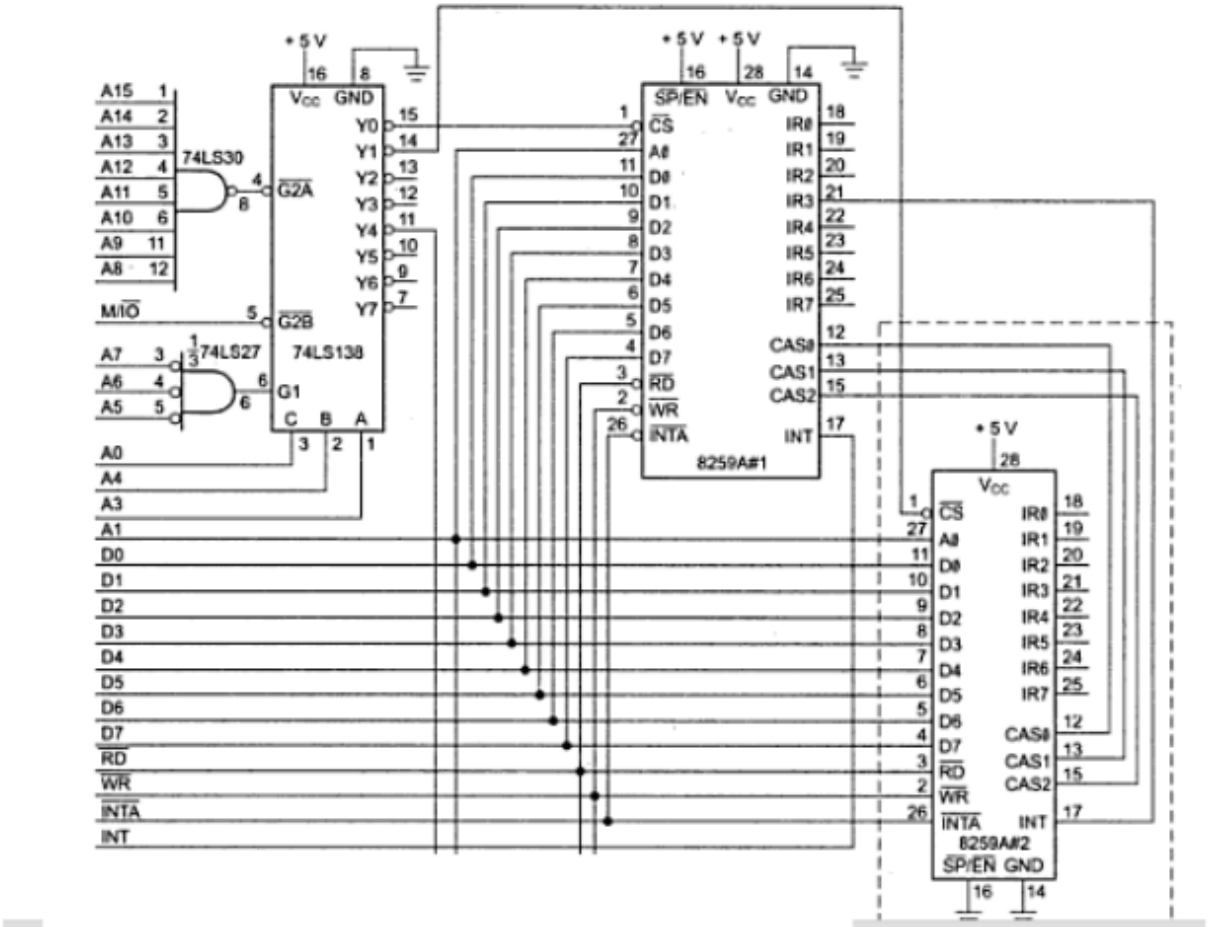
8259 Block Diagram



Multiple Interrupts



8259 System Connection



Initialization Command Word(ICW)

ICW_1

A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	**A_7**	**A_6**	**A_5**	**1**	**LITM**	**ADI**	**SNGL**	**IC_4**

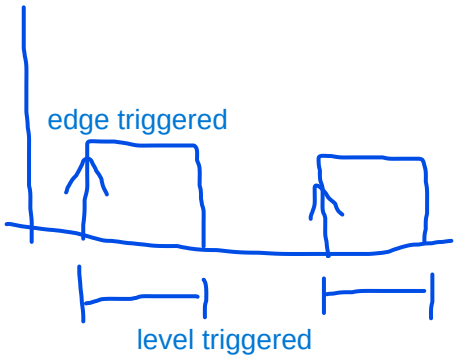
D_0 1=ICW_4 Needed
0=No ICW_4 Needed

D_1 1=Single
0=Cascaded

D_2 Call Address Interval
1=Interval of 4 bytes
0=Interval of 8 bytes

D_3 1=Level Triggered
0=Edge Triggered

A_7-A_5 of Interrupt vector address
MCs 80/85 mode only



Initialization Command Word(ICW)

ICW_2

A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	**T_7**	**T_6**	**T_5**	**T_4**	**T_3**	**A_{10}**	**A_9**	**A_8**

T_7-T_3 are $A_3 - A_0$ of Interrupt vector address
$A_{10} - A_9$, A_8 – Selected according to Interrupt request level.
They are not the address lines to microprocessor
A_0 – 1 Selects ICW_2

Fig1.4. Initialisation Command Words ICW_1 and ICW_2

Initialization Command Word(ICW)

Master ICW_3

A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0

$S_n = 1 - IR_n$ Input has a slave

$= 0 - IR_n$ Input does not have a slave

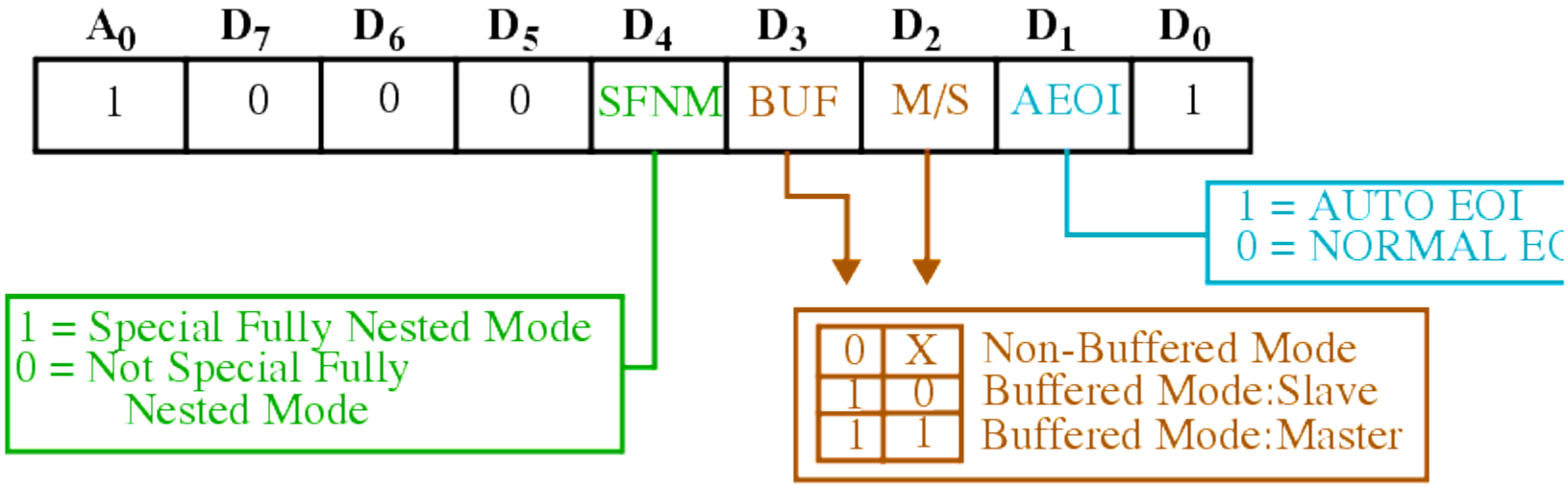
Slave ICW_4

A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	0	0	ID_2	ID_1	ID_0

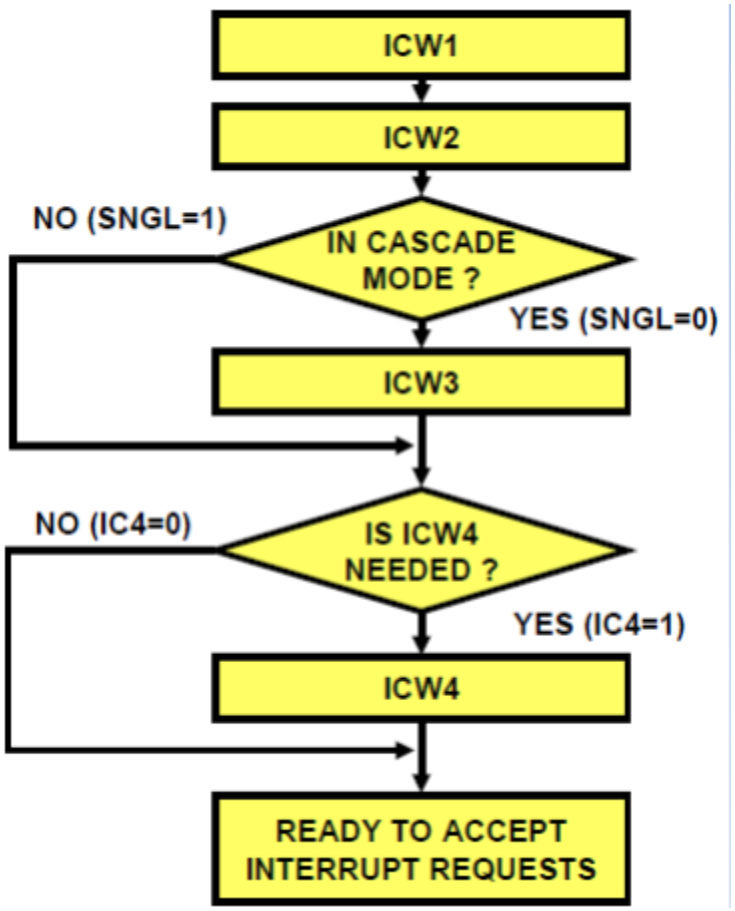
$D_2\ D_1\ D_0$ – 000 to 111 for IR_0 to IR_7 or slave 1 to slave 8

Fig1.5. ICW_3 in Master and Slave Mode

Initialization Command Word(ICW)



Initialization Sequence



Operation Command Word (OCW)

OCW_1

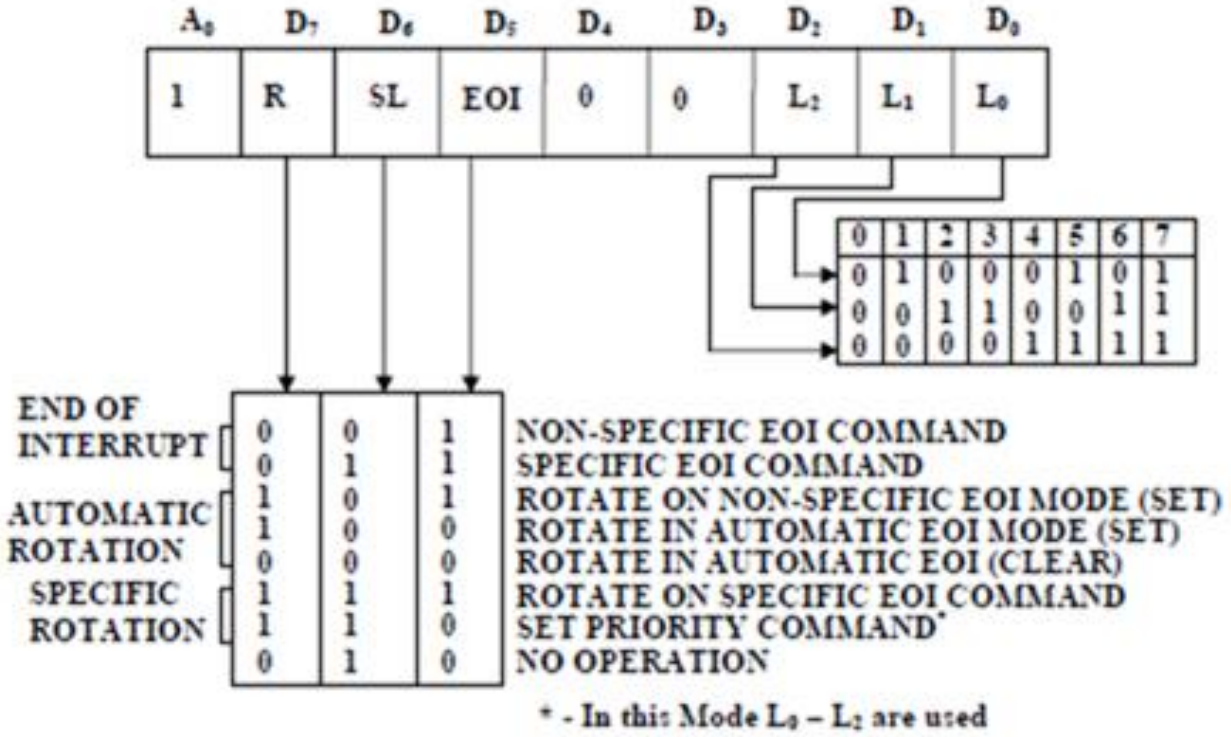
A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	**M_7**	**M_6**	**M_5**	**M_4**	**M_3**	**M_2**	**M_1**	**M_0**

1 - Mask Set

0 - Mask Clear

Operation Command Word (OCW)

OCW 2



Operation Command Word (OCW)

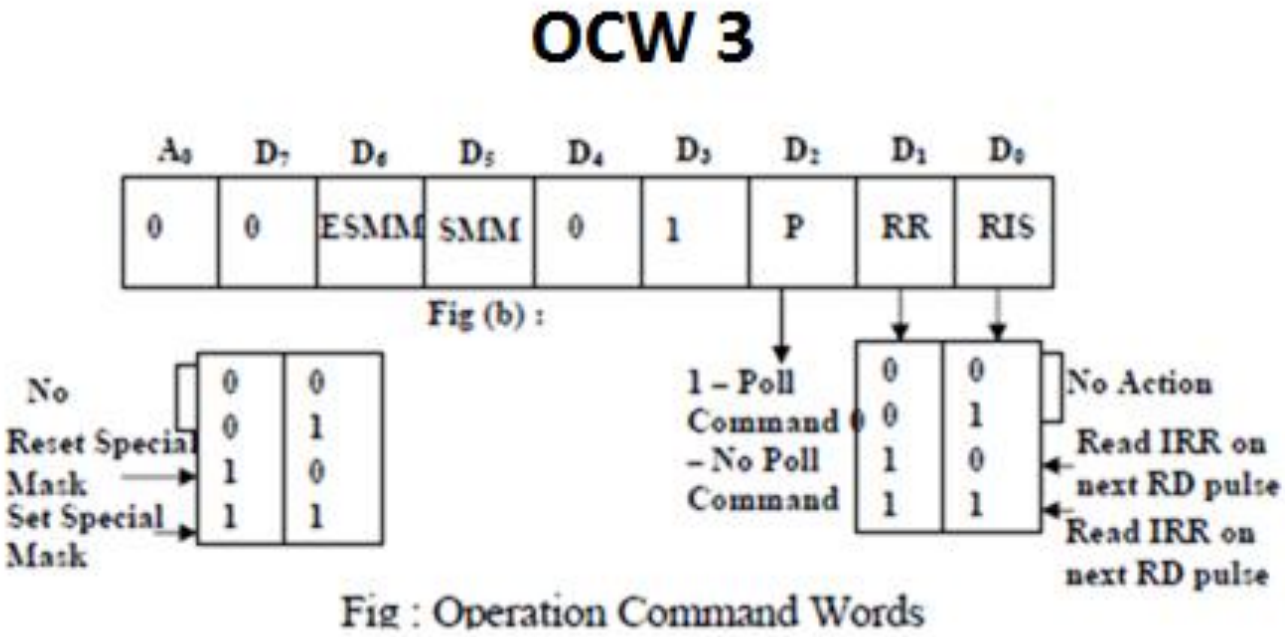


Fig : Operation Command Words