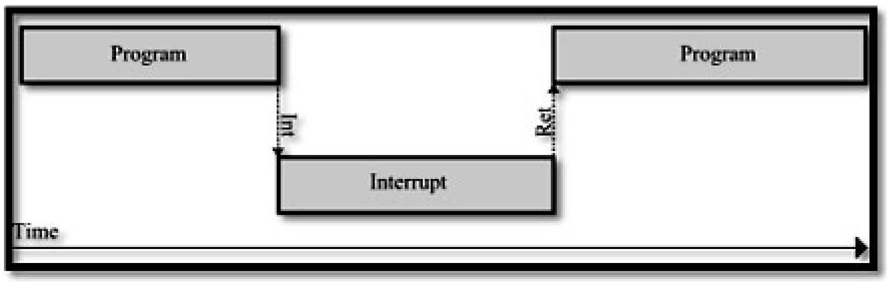
Module III

Interrupts in 8086

While the CPU is executing a program, an interrupt breaks the normal execution of instructions, diverts its execution to some other program called Interrupt Service Routine (ISR) Whenever an interrupt occurs the processor completes the execution of the current instruction and starts the execution of an Interrupt Service Routine (ISR) or Interrupt Handler.

ISR is a program that tells the processor what to do when the interrupt occurs. At the end of the ISR the last instruction should be IRET. After the execution of ISR, control returns back to the main routine where it was interrupted.



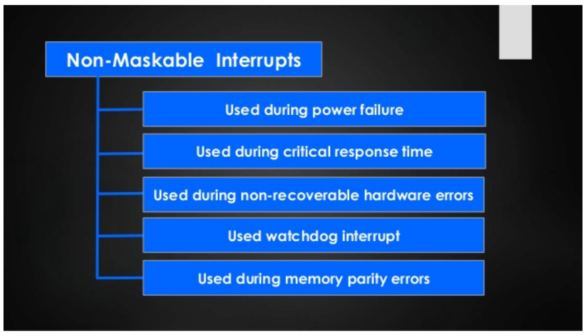
- Whenever a number of devices interrupt a CPU at a time, and if the processor is able to handle them properly, it is said to have multiple interrupt processing capability.
- There are two interrupt pins in 8086. **NMI** and **INTR**

Need for Interrupt: Interrupts are particularly useful when interfacing I/O devices that provide or require data at relatively low data transfer rate.

NMI

It is a single non-maskable interrupt pin (NMI) having higher prority. 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.(Type 2*4=00008 H)
- CS is loaded from the contents of the next word location 0000AH.
- Interrupt flag and trap flag are reset to 0.


Non-Maskable Interrupts
Used during power failure
Used during critical response time
Used during non-recoverable hardware errors
Used watchdog interrupt
Used during memory parity errors

INTR

The INTR is a maskable interrupt pin. It can be accepted (enable) or rejected (masked).

The microprocessor enabled the interrupt using set interrupt flag instruction. It should disable using clear interrupt Flag instruction.

These actions are taken by the microprocessor −

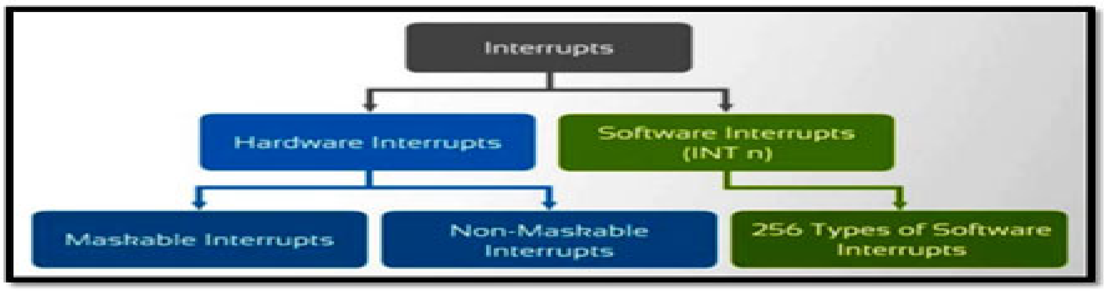
- First completes the current instruction.
- Activates INTA output and receives the interrupt type, say X.
- Flag register value, CS value of the return address and IP value of the return address are pushed on to the stack.
- IP value is loaded from the contents of word location X × 4
- CS is loaded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

Types of Interrupts

In general there are two types of Interrupts:

Internal (or) Software Interrupts are generated by a software instruction and operate similarly to a jump or branch instruction.

External (or) Hardware Interrupts are caused by an external hardware module.



HARDWARE INTERRUPTS

Hardware interrupts are generated by hardware devices when something unusual happens; this could be a key-press or a mouse move or any other action.

It can be divided into two

1. Maskable 2. Non maskable

Maskable Interrupts:

There are some interrupts which can be masked (disabled)or enabled by the processor.

Non-Maskable Interrupts:

There are some interrupts which cannot be masked out or ignored by the processor. These are associated with high priority tasks which cannot be ignored (like memory parity or bus faults).

SOFTWARE INTERRUPTS

Interrupts are generated by a software instruction and operate similarly to a jump or branch instruction.

- **256 interrupts are there**

  INT n is invoked as software interrupts- n is the type no in the range 0 to 255(00 to FF)

  Interrupts are divided into three groups

Type 0 to Type4 (Dedicated Interrupts)

- **TYPE 0** interrupt represents division by zero situation.
- **TYPE 1** interrupt represents single-step execution during the debugging of a program.
- **TYPE 2** interrupt represents non-maskable NMI interrupt.
- **TYPE 3** interrupt represents break-point interrupt.
- **TYPE 4** interrupt represents overflow interrupt.

Type 5 to 31(Not used by 8086, reserved for higher processor like 80286,80386….

Type 32-255(Available for user)

- User defined interrupts

Interrupt Service Routine

For every interrupt, there must be an interrupt service routine (ISR), or interrupt handler. When an interrupt is invoked, the microprocessor runs the interrupt service routine. For every interrupt, there is a fixed location in memory that holds the address of its ISR. The group of memory locations set aside to hold the addresses of ISRs is called the **interrupt vector table.**

When an interrupt is occurred, the microprocessor stops execution of current instruction. It transfers the content of program counter (CS and IP) into stack.

After this, it jumps to the memory location specified by Interrupt Vector Table (IVT). After that the code written on that memory area will execute.

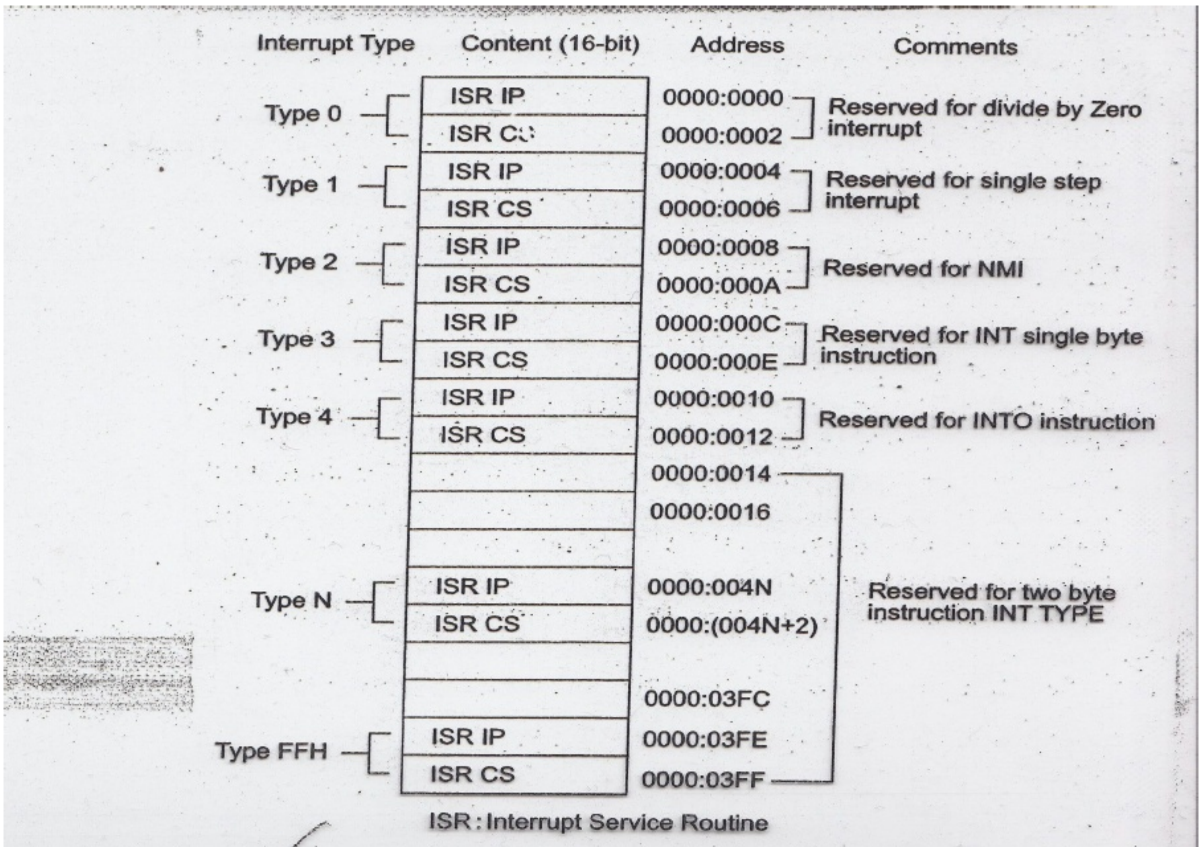


Fig. 4.5 *Structure of Interrupt Vector Table of 8086/88*

The first 1Kbyte of memory of 8086 (00000 to003FF) is set aside as a table for storing the starting addresses of Interrupt Service Procedures(ISP).Since 4-bytes are required for storing starting addresses of ISPs, the table can hold 256 Interrupt procedures.

The starting address of an ISP is often called the **Interrupt Vector or Interrupt Pointer .**Therefore the table is referred as **Interrupt Vector Table.**

NMI (Non mask-able interrupt)-

- This is a non-mask-able, edge triggered, high priority interrupt.
- On receiving an interrupt on NMI line, the microprocessor executes INT
- Microprocessor obtains the ISR address from location 2 x 4 = 00008H from the IVT.
- It reads 4 locations starting from this address to get the values for IP and CS to execute the ISR.

INTR-

- This is a mask-able, level triggered, low priority interrupt.
- On receiving an interrupt on INTR line, the microprocessor executes 2 INTA pulses.
- 1st INTA pulse – The interrupting device calculates (prepares to send) the vector number. 2nd INTA pulse – The interrupting device sends the vector number 'N' to the microprocessor.
- Now microprocessor multiplies N x 4 and goes to the corresponding location in the IVT to obtain the ISR address. INTR is a mask-able interrupt.
- It is masked by making IF = 0 by software through CLI instruction.
- It is unmasked by making IF = 1 by software through STI instruction.

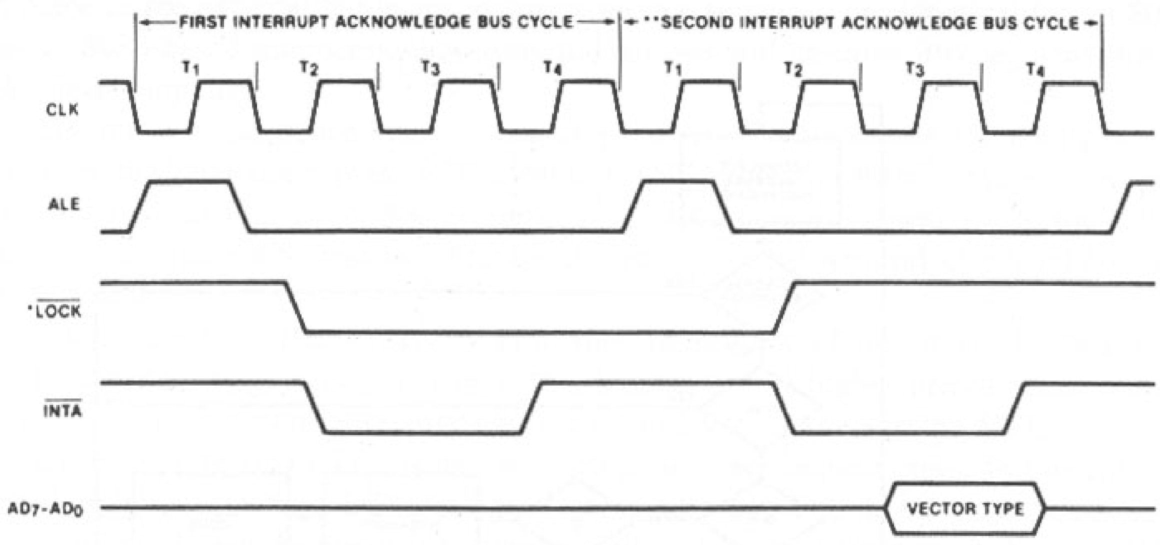

FIRST INTERRUPT ACKNOWLEDGE BUS CYCLE
**SECOND INTERRUPT ACKNOWLEDGE BUS CYCLE
T1
T2
T3
T4
CLK
ALE
*LOCK
INTA
AD7-AD0
VECTOR TYPE

Figure 11–9 Interrupt-acknowledge bus cycle. (Reprinted by permission of Intel Corporation. Copyright/Intel Corp. 1979)

Suppose an external signal interrupts the processor and the pin $\overline{LOCK}$ goes low at the trailing edge of the first ALE pulse that appears after the interrupt signal preventing the use of bus for any other purpose.

The pin $\overline{LOCK}$ remains low till the start of the next machine cycle.

With the trailing edge of $\overline{LOCK}$, the $\overline{INTA}$ goes low and remains low for two clock states before returning back to the high state.

It remains high till the start of the next machine cycle, i.e. next trailing edge of ALE.

Then $\overline{INTA}$ again goes low, remains low for two states before returning to the high state.The first trailing edge of ALE floats the bus AD_0-AD_7, while the second trailing edge prepares the bus to accept the type of the interrupt. The type of the interrupt remains on the bus for a period of two cycles.

The Operation of an Interrupt sequence on the 8086 Microprocessor:

1. External interface sends an interrupt signal, to the Interrupt Request (INTR) pin, or an internal interrupt occurs.
2. The CPU finishes the present instruction (for a hardware interrupt) and sends Interrupt Acknowledge (INTA) to hardware interface.

3. The interrupt type N is sent to the Central Processor Unit (CPU) via the Data bus from the hardware interface.

4. The contents of the flag registers are pushed onto the stack.

5. Both the interrupt (IF) and (TF) flags are cleared. This disables the INTR pin and the trap or single-step feature.

6. The contents of the code segment register (CS) are pushed onto the Stack.

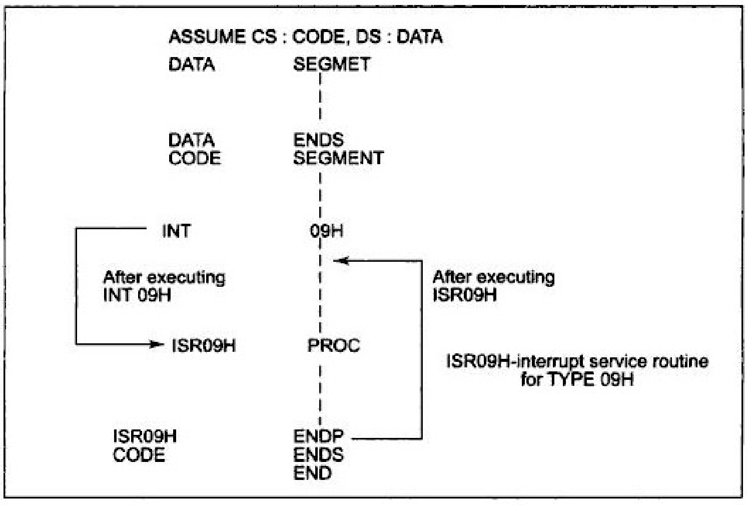
7. The contents of the instruction pointer (IP) are pushed onto the Stack.

8. The interrupt vector contents are fetched, from (4 x N) and then placed into the IP and from (4 x N +2) into the CS so that the next instruction executes at the interrupt service procedure addressed by the interrupt vector.

9. While returning from the interrupt-service routine by the Interrupt Return (IRET) instruction, the IP, CS and Flag registers are popped from the Stack and return to their state prior to the interrupt.

Interrupt Programming

While programming for any type of interrupt, the programmer must, either externally or through the program, set the interrupt vector table for that type preferrably with the CS and IP addresses of the interrupt service routine. The method of defining the interrupt service routine for software as well as hardware interrupt is the same. Figure 4.7 shows the execution sequence in case of a software interrupt.



```
ASSUME CS : CODE, DS : DATA
DATA      SEGMET
          |
          |
DATA      ENDS
CODE      SEGMENT
          |
          |
INT       09H
          |
          |
          |
ISR09H    PROC
          |
          |
ISR09H    ENDP
CODE      ENDS
          END
```

Fig. 4.7 *Transfer of Control during Execution of an Interrupt Service Routine*

Program

Write a program to a create a file RESULT and store in it 500H bytes from the memory block starting at 1000:1000, if either an interrupt occurs at INTR pin with Type 0AH or an instruction equivalent to the above interrupt is executed.

```
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 ; Set DS at code for setting IVT
MOV DX,OFFSET ISR0A ; Set DX at offset of ISR0A.
MOV AX,250AH ;Set IVT using function value 250AH
INT 21H ; in AX under INT 21H
MOV DX,OFFSET FILENAME ; Set pointer to filename.
MOV AX,DATA ; Set the DS at DATA for filename.
MOV DS,AX
MOV CX,00H
MOV AH,3CH ; Create file with the filename ‘RESULT’
INT 21H
JNC FURTHER ; If no carry, create operation is successful
MOV DX,OFFSET MESSAGE ; else display message
MOV AH,09H
INT 21H
JMP STOP
FURTHER: INT 0AH ; If the file is created successfully,
MOV AH,4CH ; write into it and return to DOS prompt
INT 21H
ISR0A PROC NEAR
MOV BX,AX ; Take file handle in BX,
MOV CX,500H ; Byte count in CX
MOV DX,1000H ; Offset of block in DX
MOV AX,1000H ; Segment value of block
MOV DS,AX ; in DS
MOV AH,40H ; Write in the file and
INT 21H ; return
ISR0AH ENDP
CODE ENDS
END START
```

INTEL 8259A Programmable Interrupt Controller

The 8259A is a programmable interrupt controller designed to work with Intel microprocessor 8080 A, 8085, 8086, 8088.

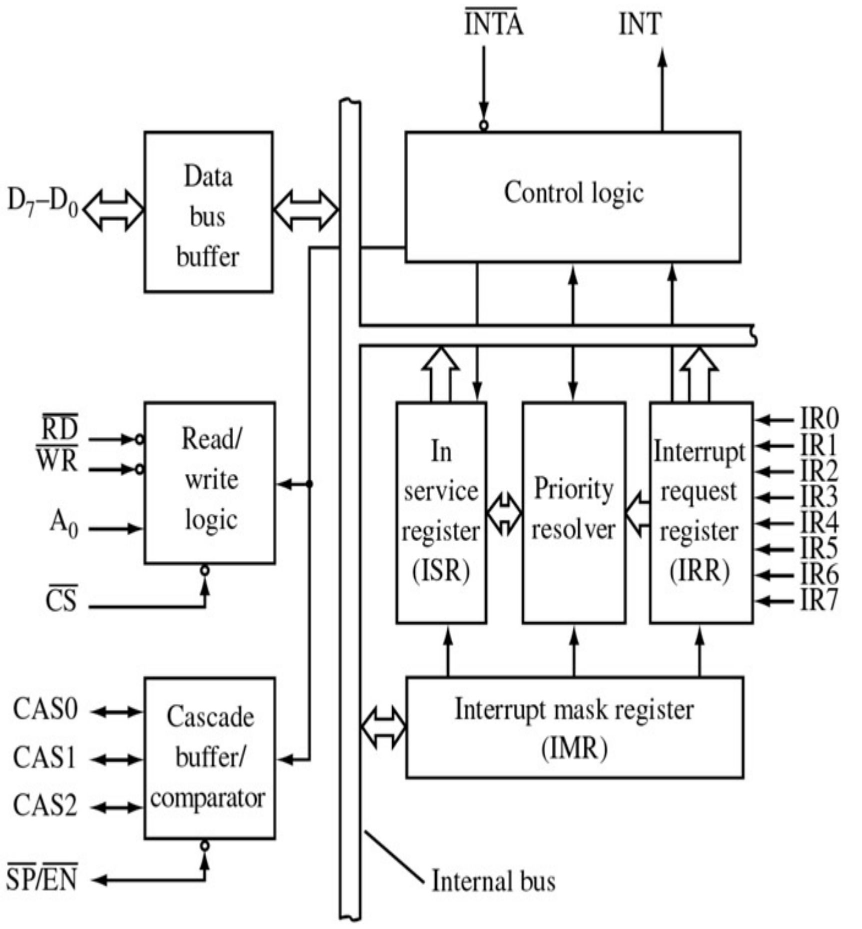
Programmable Interrupt controller able to handle no of interrupts at a time.

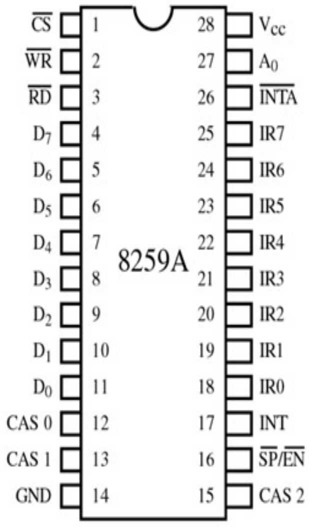
Features:

- 8 levels of interrupts.
- Can be cascaded in master-slave configuration to handle 64 levels of interrupts.
- Internal priority resolver.
- Individually maskable interrupts.

- Modes and masks can be changed dynamically.
- Accepts IRQ, determines priority, checks whether incoming priority > current level being serviced, issues interrupt signal.
- In 8086 mode, provides 8 bit vector number.
- Starting address of ISR or vector number is programmable.
- No clock required.

Architecture of 8259A





Interrupt request register (IRR):IRR stores all the interrupt inputs that are requesting service. Basically, it keeps track of which interrupt inputs are asking for service. If an interrupt input is unmasked (enabled), and has an interrupt signal on it, then the corresponding bit in the IRR will be set.

- The IRR is used to indicate all the interrupt levels which are requesting service, and the ISR is used to store all the interrupt levels which are currently being serviced

In service register (ISR):

The in service registers keeps tracks of which interrupt inputs are currently being serviced. For each input that is currently being serviced the corresponding bit will be set in the in service register.

Priority Resolver:

This logic block determines the priorities of the set in the IRR. The highest priority is selected and stored into the corresponding bit of the ISR during pulse.

Interrupt mask register (IMR):

The IMR stores the bits which disable the interrupt lines to be masked. The IMR operates on the output of the IRR

Priority Resolver:

This logic block determines the priorities of the set in the IRR. The highest priority is selected and stored into the corresponding bit of the ISR during pulse.

Data bus buffer:

This 3- state, bidirectional 8-bit buffer is used to interface the 8259A to the system data bus. Control words and status information are transferred through the data bus buffer.

Read/Write & control logic:

The function of this block is to accept and decodes commands from the CPU. This function block also allows the status of 8159A to be transferred to the data bus.

Cascade buffer/comparator:

This function blocks stores and compare the IDS of all 8259A's in the reg. The associated 3-I/O pins (CAS0-CAS2) are outputs when 8259A is used a master and are inputs when 8259A is used as a slave. As a master, the 8259A sends the ID of the interrupting slave device onto the CAS2-CAS0. The slave thus selected will send its pre-programmed subroutine address on to the data bus during the next one or two successive pulses.

Pin Description

The 8259 A is contained in a 28-element in line package

The pins are defined as follows:

D0-D7	Bi-directional, tristated, buffered data lines. Connected to data bus directly or through buffers
RD-bar	Active low read control
WR-bar	Active low write control
A0	Address input line, used to select control register
CS-bar	Active low chip select
CAS0-2	Bi-directional, 3 bit cascade lines. In master mode, PIC places slave ID no. on these lines. In slave mode, the PIC reads slave ID no. from master on these lines.
SP-bar / EN-bar	Slave program / enable. In non-buffered mode, it is SP-bar input, used to distinguish master/slave PIC. In buffered mode, it is output line used to enable buffers
INT	Interrupt line, connected to INTR of microprocessor
INTA-bar	Interrupt ack, received active low from microprocessor
IR0-7	Asynchronous IRQ input lines, generated by peripherals.

Interrupt Sequence in an 8086 system

The Interrupt sequence in an 8086-8259A system is described as follows:

1. 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 (AEOI) mode is programmed. Otherwise ISR bit remains set until an appropriate EOI command is issued at the end of interrupt subroutine.

Command Words of 8259A

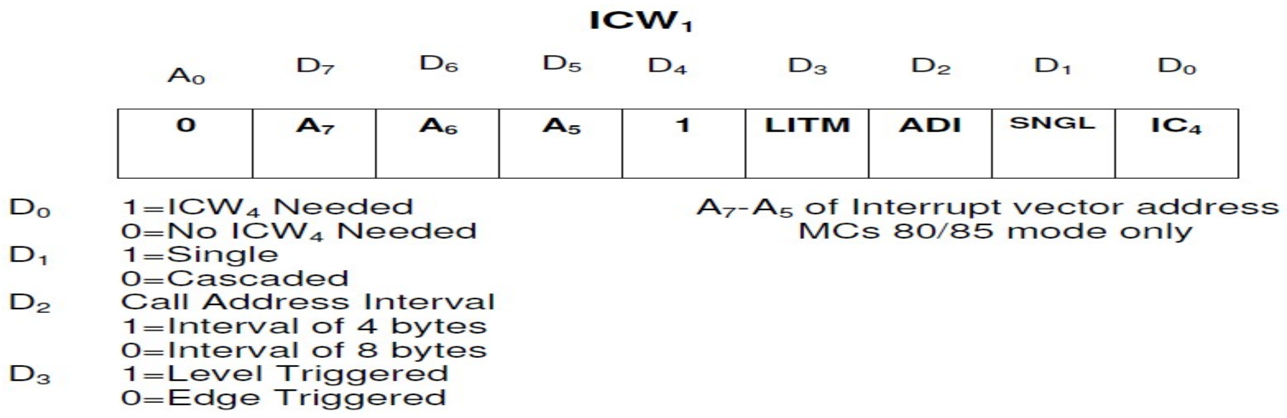
The command words of 8259A are classified in two groups, viz. initialization command words (ICWs) and operation command words (OCWs)

Initialization Command Words (ICWs)

Before it starts functioning, the 8259A must be initialized by writing **two to four command** words into the respective command word registers. These are called as initialization command words (ICWs).

- ICW- contains 9 bit values. The least significant 8 bits are sent on the PC's data-bus, while the 9th bit is sent as bit 0 on the PC's address-bus

ICW1 and ICW2 are compulsory command words in initialization sequence of 8259A while ICW3 and ICW4 are optional.



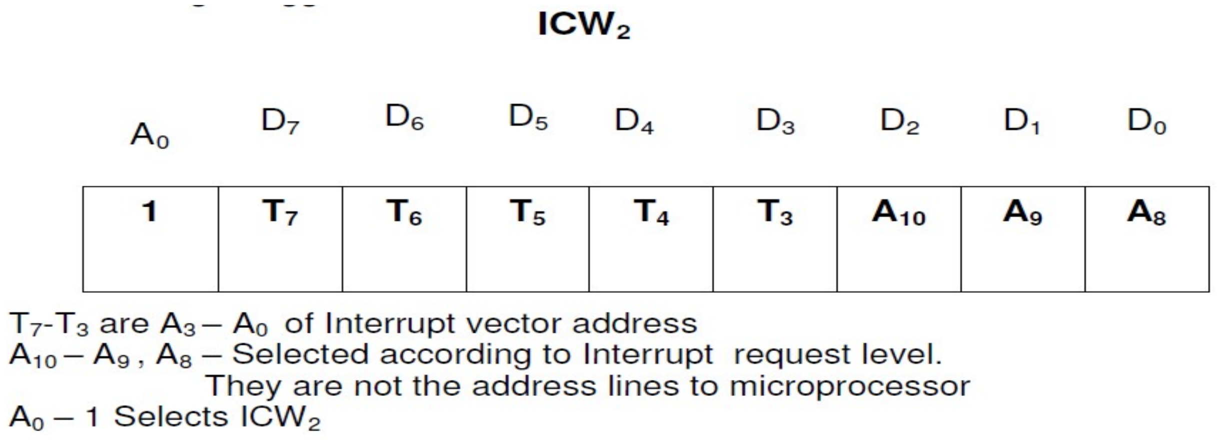
ICW_1

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

If **A0 = 0 and D4 = 1**, the control word is recognized as **ICW1**

It contains the control bits for edge/level triggered mode, single/cascade mode, call address interval and whether ICW4 is required or not, etc.



ICW_2

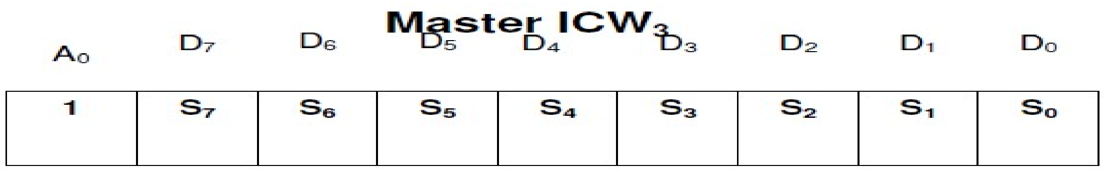
A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	T_7	T_6	T_5	T_4	T_3	A_{10}	A_9	A_8

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

If A0 = 1, the control word is recognized as **ICW2**. The ICW2 stores details regarding interrupt vector addresses.

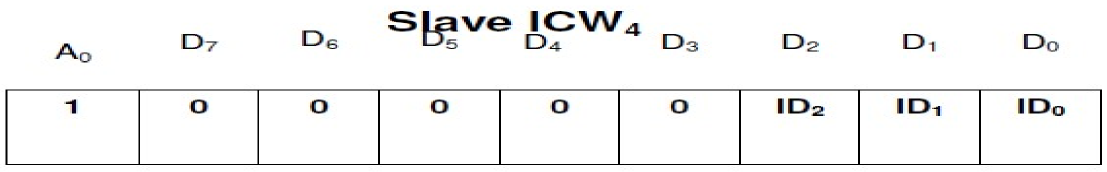
The ICW3 is read only when there are more than one 8259As in the system, i.e. cascading is used (SNGL = 0). The SNGL bit in ICW1 indicates whether the 8259A is in cascade mode or not. The ICW3 loads an 8-bit slave register.



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

ICW4 The use of this command word depends on the IC4 bit of ICW1. If IC4= 1, ICW4 is used, otherwise it is neglected. The bit functions of ICW4 are described as follows:

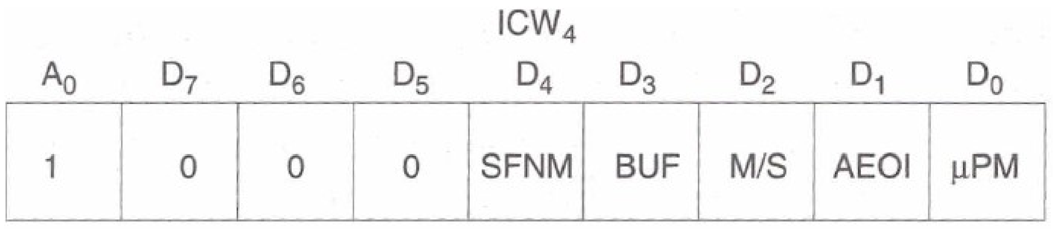
SFNM Special fully nested mode is selected, if SFNM = 1.

BUF If BUF = 1, the buffered mode is selected. In the buffered mode,SP/EN acts as enable output and the master/slave is determined using the M/S bit of ICW4.

M/S If M/S = 1, 8259A is a master. If M/S = 0, 8259A is a slave. If BUF =0, M/S is to be neglected.

AEOI: If AEOI = 1, the automatic end of interrupt mode is selected.

µPM If the µPM bit is 0, the Mcs-85 system operation is selected and if/µPM =1, 8086/88 operation is selected.



ICW_4

A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	SFNM	BUF	M/S	AEOI	µPM

Fig1.6 shows the ICW_4 bit positions

Operation Command Words

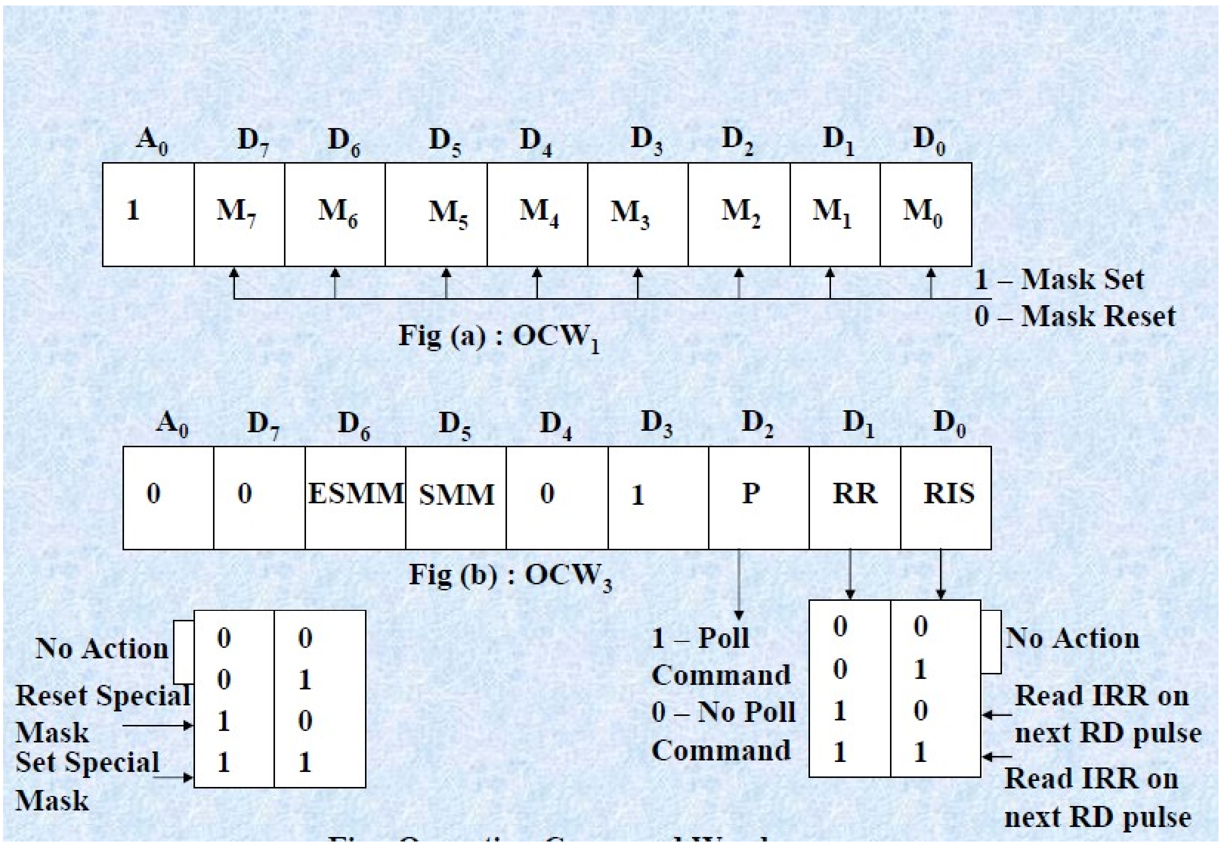
Once 8259A is initialized using the previously discussed command words for initialization, it is ready for its normal function, i.e. for accepting the interrupts but 8259A has its own ways of handling the received

interrupts called as modes of operation. These modes of operations can be selected by programming, i.e. writing three internal registers called as operation command word registers.
The data written into them (bit pattern) is called as **operation command words**.
In the three operation command words OCWl ,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 is used to mask the unwanted interrupt requests. If the mask bit is '1', the corresponding interrupt request is masked, and if it is '0', the request is enabled.
In OCW2 the three bits, viz. R, SL and EOI control the end of interrupt, the rotate mode and their combinations as shown in Fig.
The three bits L2, L1 and L0 in OCW2 determine the interrupt level to be selected for operation , if the SL bit is active, i.e. '1'. The details of OCW2 are shown in Fig.

In operation command word 3 (OCW3), if the ESMM bit, i.e. enable special mask mode bit is set to '1', the SMM bit is enabled to select or mask the special mask mode. When ESMM bit is '0', the SMM bit is neglected. If the SMM bit, i.e.special mask mode bit is '1', the 8259A will enter special mask mode provided ESMM = 1.

If ESMM = 1 and SMM = 0, the 8259A will return to the normal mask mode.


A0 D7 D6 D5 D4 D3 D2 D1 D0
1 M7 M6 M5 M4 M3 M2 M1 M0
1 – Mask Set
0 – Mask Reset
Fig (a) : OCW1
A0 D7 D6 D5 D4 D3 D2 D1 D0
0 0 ESMM SMM 0 1 P RR RIS
Fig (b) : OCW3
No Action
Reset Special Mask
Set Special Mask
0 0
0 1
1 0
1 1
1 – Poll Command
0 – No Poll Command
0 0
0 1
1 0
1 1
No Action
Read IRR on next RD pulse
Read IRR on next RD pulse

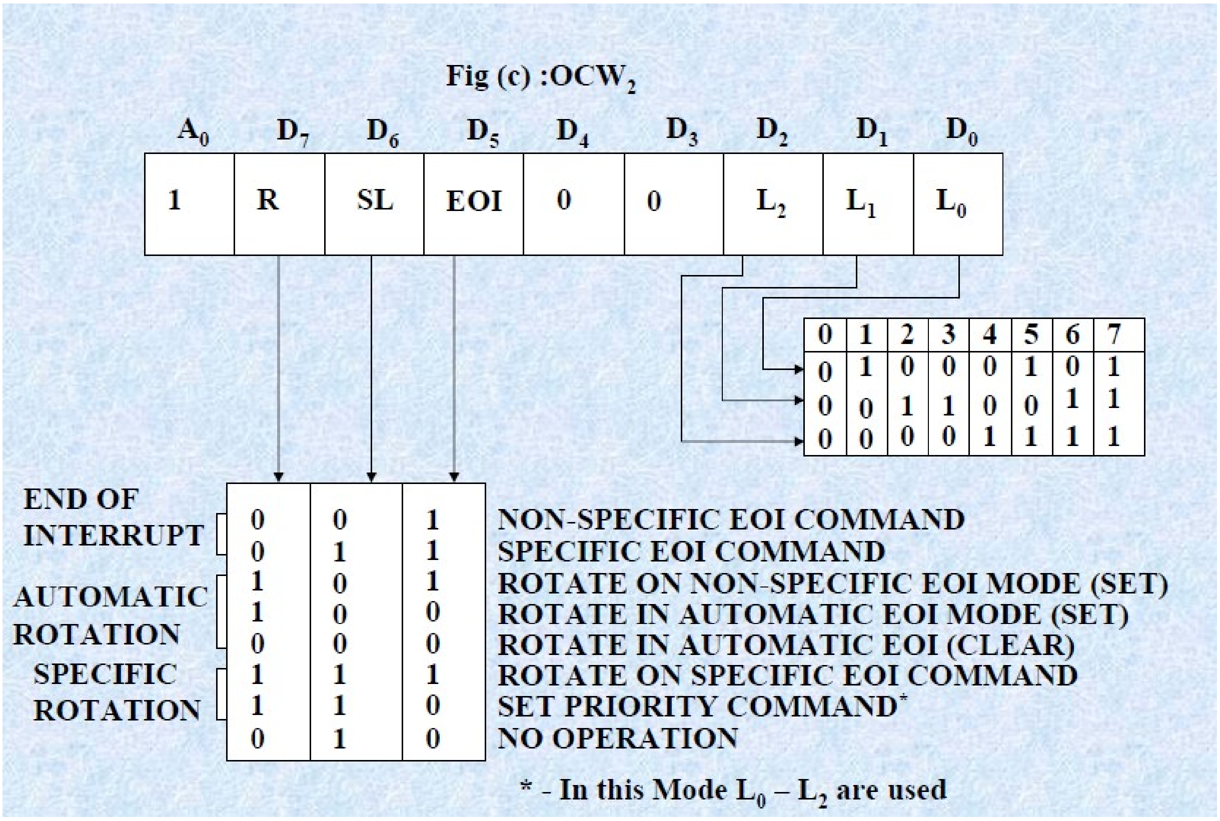


Fig (c) :OCW$_2$
A_0 D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0
1 R SL EOI 0 0 L_2 L_1 L_0
0 1 2 3 4 5 6 7
END OF INTERRUPT
AUTOMATIC ROTATION
SPECIFIC ROTATION
NON-SPECIFIC EOI COMMAND
SPECIFIC EOI COMMAND
ROTATE ON NON-SPECIFIC EOI MODE (SET)
ROTATE IN AUTOMATIC EOI MODE (SET)
ROTATE IN AUTOMATIC EOI (CLEAR)
ROTATE ON SPECIFIC EOI COMMAND
SET PRIORITY COMMAND*
NO OPERATION
* - In this Mode $L_0 - L_2$ are used

R	SL	EOI	
0	0	1	NON-SPECIFIC EOI COMMAND
0	1	1	SPECIFIC EOI COMMAND
1	0	1	ROTATE ON NON-SPECIFIC EOI MODE (SET)
1	0	0	ROTATE IN AUTOMATIC EOI MODE (SET)
0	0	0	ROTATE IN AUTOMATIC EOI (CLEAR)
1	1	1	ROTATE ON SPECIFIC EOI COMMAND
1	1	0	SET PRIORITY COMMAND*
0	1	0	NO OPERATION

	0	1	2	3	4	5	6	7
L_0	0	1	0	0	0	1	0	1
L_1	0	0	1	1	0	0	1	1
L_2	0	0	0	0	1	1	1	1

Operating Modes of 8259

Fully Nested Modes

.Default mode.

.IR0 has the highest priority and IR7 has the lowest one.

.If the ISR (in service) bit is set, all the same or lower priority interrupts are inhibited.

2.End Of Interrupt

.The ISR bit can be reset either with AEOI bit of ICW1 or by EOI command.

.Two types of EOI command;

a)Specific

b)Non-specific

.The non-specific EOI command automatically reset the highest ISR bit.

.When a mode that may disturb the fully nested structure, the specific EOI command is issued to reset a particular ISR bit.

.An ISR bit that is masked by the corresponding IMR bit, will not be cleared by a non-specific EOI , if it is in special mode.

3.<u>Automatic Rotation</u>

.Used in the applications where , all the interrupting devices are of equal priority.

.In this mode, an IR level receives lowest priority after it is served while the next device to be served gets the highest priority in sequence.

4.<u>Automatic EOI Mode</u>

.Till AEOI=1 in ICW4, 8259A operates in AEOI mode.

.The 8259A performs a non- specific EOI at the trailing edge of the last /INTA pulse automatically.

.AEOI should be used only when a nested multilevel interrupt structure is not required.

5.<u>Specific Rotation</u>

.A bottom priority level can be selected, using L2, L1 and L0 inOCW2 and R=1, SL=1,EOI=0.

.The selected bottom priority fixes other priorities.

6.<u>Special Mask Mode</u>

.When a mask bit is set in OCW, it inhibits further interrupts at that level and enables interrupt from other levels, which are not mastered.

7.<u>Edge And Level Triggered Mode</u>

.Decides whether the interrupt should be edge triggered or level triggered.

.If bit LTIM of ICW1=0, they are edge triggered, otherwise level triggered.

8.<u>Readng 8259A Status</u>

.Used to read the , status of the internal registers of 8259A.

.Reading is possible only in no polled mode.

.OCW3, is used to read IRR and ISR and OCW1 for IMR.

9.<u>Poll command</u>

.The INT output is neglected, though it functions normally by not connecting INT output or by masking INT input of the microprocessor.

.This mode is entered by setting p=1 in OCW3.

.A poll command may give more than 64 priority levels.

10.<u>Special Fully Nested Mode</u>

.Used in more complicated systems.

.Similar to, normal nested mode.

.When an interrupt request from a certain slave is in service, this slave can further send requests to the master.

.The master interrupts the CPU only.

11.Buffered Mode

.When the 8259A is used in the system in which bus driving buffers are used on the data buses, the problem of enabling the buffers arises.

.The 8259A sends a buffer enable signal on /SP//EN pin.

12.Cascade Mode

.The slave INT outputs are connected with master IR inputs. When a slave request line is activated and acknowledged, the master will enable the slave to release the vector addresses during the second pulses of INTA sequence.

.The cascade lines are normally low and contain slave addresses codes from the trailing edge of the first INTA pulse to the trailing edge of the second /INTA pulse.

Interfacing 8259 with 8086

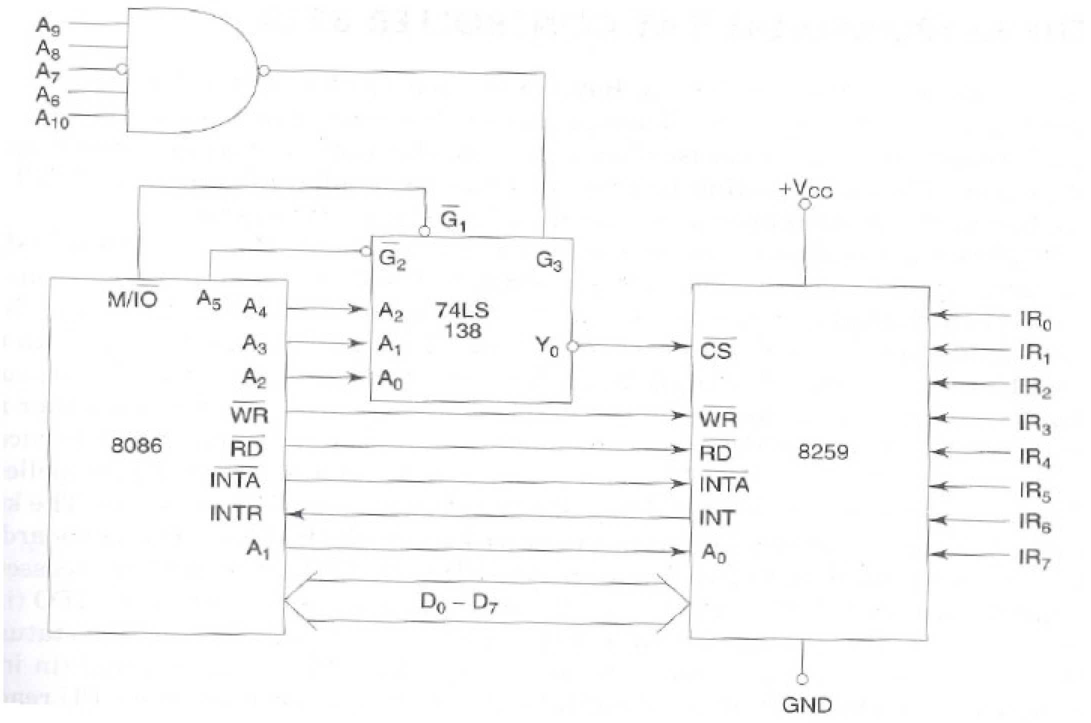


Fig1.10. Interfacing 8259A with 8086