COMPUTER ENGINEERING DEPARTMENT

ASSIGNMENT NO-01

SUB: Microprocessor

COURSE: T.E. **Year: 2020-2021** **Semester: V**

DEPT: Computer Engineering

SUBJECT CODE: CSC501 **SUBMISSION DATE: 26/11/2020**

==========

Name: Amey Thakur **Roll No.:** 50

Batch: B3 **Class:** TE COMPS B

Assignment I

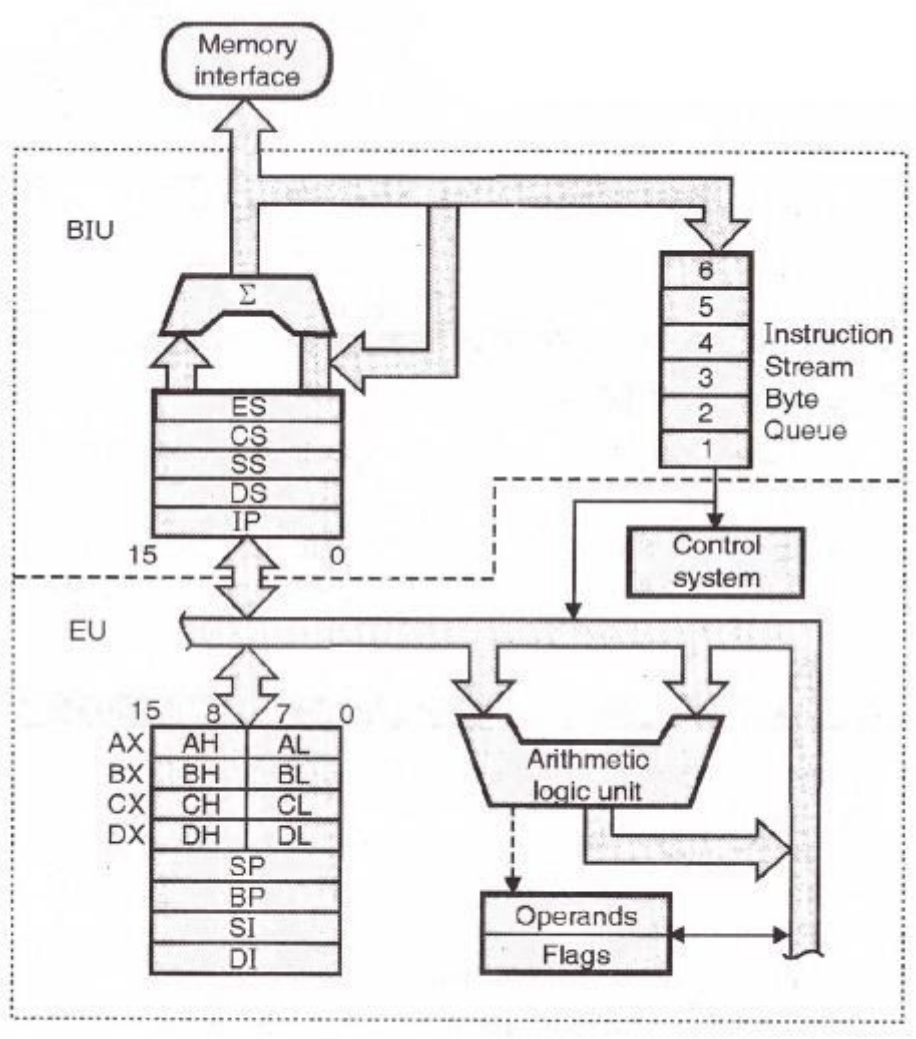
Q. No	Questions	CO Mapping	PO Mapping
1.	Give a brief description of the architecture of the 8086 processor.	1	1,2,3,4,5,8, 10
2.	What is segmentation and its advantages?	1	1,2,3,4,5,8, 10
3.	Write an assembly language program to display string in lowercase.	2	1,2,3,4,5,8, 10,11,12
4.	Write an assembly language program to print the factorial of a number.	2	1,2,3,4,5,8, 10,11,12
5.	What is an interrupt to the processor? Explain IVT.	3	1,2,3,4,5,8, 10,11,12
6.	Describe PIC block diagram with its operating mode.	4	1,2,3,4,5,8, 10,11,12

1. **Give a brief description of the architecture of the 8086 processor.**

Ans:

As 8086 does 2-stage pipelining (overlapping fetching and execution), its architecture is divided into two units:

1. **Bus Interfacing Unit (BIU)**
2. **Execution Unit (EU)**



8086 internal architecture

Bus Interfacing Unit (BIU)-

- ➔ It provides the interface of 8086 to external memory and I/O devices.
- ➔ It operates with respect to bus cycles (machine cycles). This means it performs various machine cycles such as memory read, I/O read etc. to transfer data with memory and I/O devices.
- ➔ BIU performs the following functions-
  - ◆ It generates the 20-bit physical address for memory access.
  - ◆ It fetches an instruction from memory.

- It transfers data to and from the memory and I/O.
- It supports pipelining using the 6-byte instruction queue.

The main components of the BIU are as follows:

➔ **Segment registers-**
- **CS register:** CS holds the base address for the Code Segment. All programs are stored in the Code Segment. CS is multiplied by 10H to give the 20-bit physical address of the Code Segment. E.g. If CS = 4321H then CS x 10H = 43210H→ Starting address of Code Segment.
- **DS register:** DS holds the base address for the Data Segment. It is multiplied by 10H to give the 20-bit physical address of the Data Segment. E.g. If DS = 4321H then DS x 10H = 43210H→ Starting address of Data Segment.
- **SS register:** SS holds the base address for the Stack Segment. It is multiplied by 10H to give the 20-bit physical address of the Stack Segment. E.g. If SS = 4321H then SS x 10H = 43210H→ Starting address of Stack Segment.
- **ES register:** ES holds the base address for the Extra Segment. It is multiplied by 10H to give the 20-bit physical address of the Extra Segment. E.g. If ES = 4321H then ES x 10H = 43210H→ Starting address of Code Segment.

➔ **Instruction Pointer (IP)-**
- It is a 16-bit register. It holds the offset of the next instructions in the Code Segment.
- Address of the next instruction is calculated as CS x 10H + IP.
- IP is incremented after every instruction byte is fetched.
- IP gets a new value whenever a branch occurs.

➔ **Address Generation Circuit-**
- The BIU has a Physical Address Generation Circuit. It generates the 20-bit physical address using Segment and Offset addresses using the formula: Physical Address = Segment Address x 10H + Offset Address

➔ **6 Byte Prefetch Queue-**
- It is a 6 byte first in first out (FIFO) RAM used to implement pipelining. Fetching the next instruction while executing the current instruction is called pipelining.
- BIU fetches the next six instruction bytes from the Code Segment and stores it into the queue.
- Execution Unit (EU) removes instructions from the queue and executes them. The queue is refilled when at least two bytes are empty as 8086 has a 16-bit data bus. Pipelining increases the efficiency of the microprocessor.
- Pipelining fails when a branch occurs as the pre-fetched instructions are no longer useful.

- Hence as soon as 8086 detects a branch operation, it clears/discards the entire queue.
- Now, the next six bytes from the new location (branch address) are fetched and stored in the queue and pipelining continues.

Execution Unit (EU)-

- ➔ It fetches instructions from the Queue in BIU, decodes and executes them.
- ➔ It performs arithmetic, logic and internal data transfer operations within the microprocessor.
- ➔ It sends request signals to the BIU to access the external module.
- ➔ It operates with respect to T-stats (clock cycles) and does not depend upon which machine cycle is being performed by the BIU.

The main components of the EU are as follows:

- ➔ **General-purpose registers**- 8086 microprocessor has four 16 bit general purpose registers AX, BX, CX and DX. These are available to the programmer for storing values during programs. Each of these can be divided into two 8 bit registers such as AH, Al; BH, BL; etc. Besides their general use, these registers also have some specific functions.
  - AX register (16 bits): It holds operands and results during multiplication and division operations. All I/O data transfers using IN and OUT instructions use A register (AL/AH or AX). It functions as an accumulator during string operations.
  - BX register (16 bits): It holds the memory address (offset address) in indirect addressing modes.
  - CX register (16 bits): It holds the count for instructions like a loop, rotates, shifts and string operations.
  - DX register (16 bits): It is used with AX to hold 32-bit values during multiplication and division. It is used to hold the address of the I/O port in indirect I/O addressing mode.
- ➔ **Special purpose registers-**
  - **Stack Pointer (SP 16 bits):** It holds the offset address of the top of the Stack. The stack is a set of memory locations operating in a LIFO manner. The stack is present in the memory in Stack Segment. It is used during instructions like PUSH, POP, CALL, RET etc.
  - **Base Pointer (BP 16 bits):** BP can hold offset addresses of any location in the stack segment. It is used to access random locations of the stack.
  - **Source Index (SI 16 bits):** It is normally used to hold the offset address for the Data Segment but can also be used for other segments using Segment Overriding. It holds the offset address of source data in the Data Segment during string operations.

- **Destination Index (DI 16 bits):** It is normally used to hold the offset address for Extra Segment but can also be used for other segments using Segment Overriding. It holds the offset address of destination in the Extra Segment during string operations.

➔ **ALU (Arithmetic Logic Unit)** - It has a 16 bit ALU. It performs 8 and 16-bit arithmetic and logic operations.

➔ **Operand register**- It is a 16-bit register used by the control register to hold the operands temporarily. It is not available to the programmer.

➔ **Instruction Register and Instruction Decoder**- The EU fetches an opcode from the queue into the instruction register. The instruction decoder decodes it and sends the information to the control circuit for execution.

➔ **Flag register (16 bits)**-

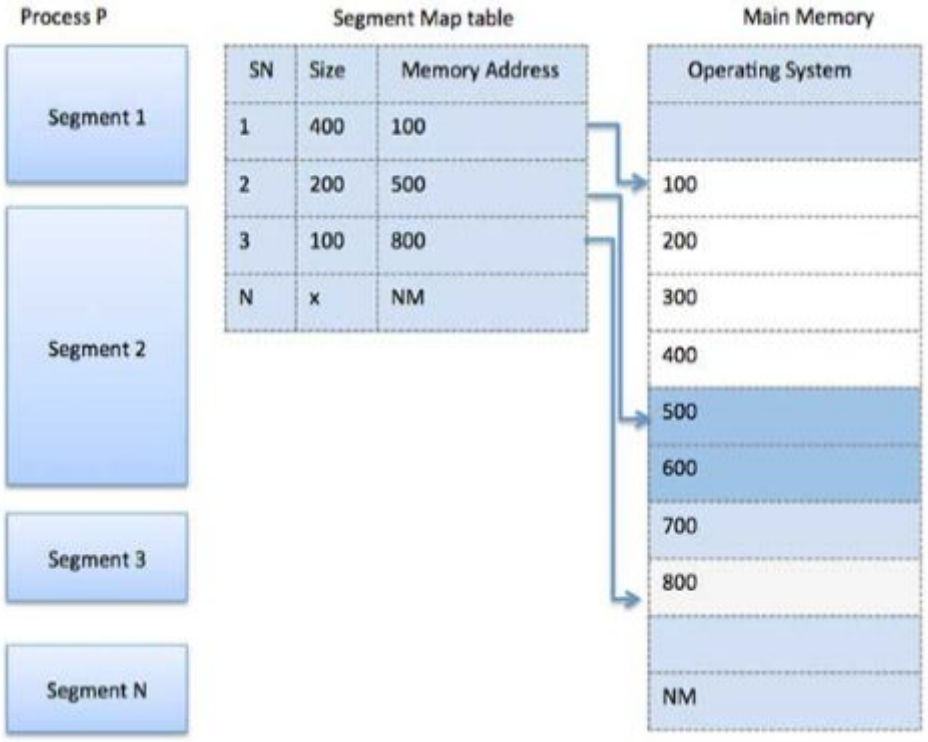
- It has 9 flags.
- These flags are of two types: 6 Status flags namely carry flag, parity flag, auxiliary carry flag, zero flag, sign flag and 3 Control flags namely trap flag, interrupt flag and direction flag.
- Status flags are affected by the ALU after every arithmetic or logic operation. They give the status of the current result.
- Control flags are used to control certain operations. They are changed by the programmer.

2. **What is segmentation and its advantages?**

Ans:

Segmentation

- ➔ Segmentation is a memory management technique in which each job is divided into several segments of different sizes, one for each module that contains pieces that perform related functions. Each segment is actually a different logical address space of the program.
- ➔ When a process is to be executed, its corresponding segmentation is loaded into non-contiguous memory though every segment is loaded into a contiguous block of available memory.
- ➔ Segmentation memory management works very similar to paging but here segments are of variable-length whereas in paging pages are of fixed size.
- ➔ A program segment contains the program's main function, utility functions, data structures, and so on. The operating system maintains a segment map table for every process and a list of free memory blocks along with segment numbers, their size and corresponding memory locations in main memory. For each segment, the table stores the starting address of the segment and the length of the segment. A reference to a memory location includes a value that identifies a segment and an offset.



Advantages of memory segmentation:

1. Segmentation provides a powerful memory management mechanism.
2. It allows programmers to partition their programs into modules that operate independently of one another.
3. Segments allow two processes to easily share data.
4. It allows the addressability of a processor i.e. segmentation allows the use of 16-bit registers to give an addressing capability of 1 MB. Without segmentation, it would require 20-bit registers.
5. Segmentation makes it possible to separate the memory areas for stack, code and data.
6. It is possible to increase the memory size of code data or stack segments beyond 64 KB by allotting more than one segment for each area.

3. **Write an assembly language program to display string in lowercase.**

Ans:

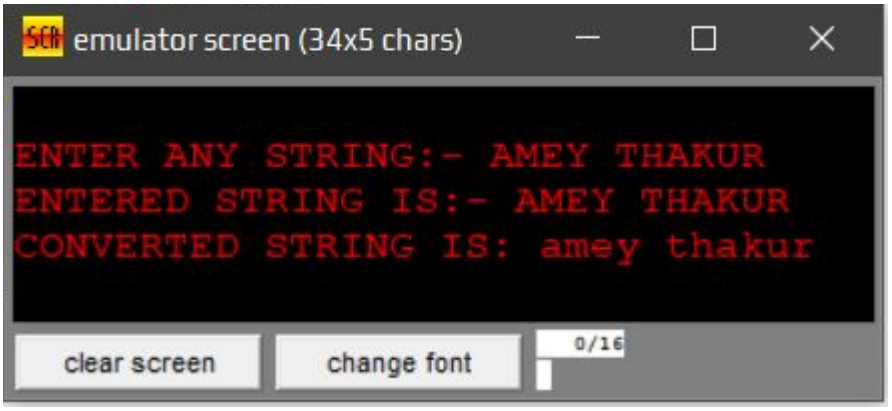
➔ **Input to display string in lowercase -**

```
DATA SEGMENT
   MSG1 DB 10,13,'ENTER ANY STRING :- $'
   MSG2 DB 10,13,'ENTERED STRING IS :- $'
   MSG3 DB 10,13,'CONVERTED STRING IS : $'

   P1 LABEL BYTE
   M1 DB 0FFH
   L1 DB ?
   P11 DB 0FFH DUP ('$')
DATA ENDS
DISPLAY MACRO MSG
   MOV AH,9
   LEA DX,MSG
   INT 21H
ENDM
CODE SEGMENT
   ASSUME CS:CODE,DS:DATA
START:
      MOV AX,DATA
      MOV DS,AX
      DISPLAY MSG1
      LEA DX,P1
      MOV AH,0AH
      INT 21H
      DISPLAY MSG2
      DISPLAY P11
      DISPLAY MSG3
      LEA SI,P11
      MOV CL,L1
      MOV CH,0
CHECK:
      CMP [SI],41H
      JB DONE
      CMP [SI],5BH
      JB LWR
      CMP [SI],61H
      JB DONE
      CMP [SI],7BH
      JG DONE
```

```
UPR:    SUB [SI],20H
        JMP DONE
LWR:    ADD [SI],20H
DONE:   INC SI
        LOOP CHECK
        DISPLAY P11
        MOV AH,4CH
        INT 21H
CODE ENDS
END START
```

➔ **Output -**



4. **Write an assembly language program to print the factorial of a number.**

Ans:

➔ **Input to find factorial of a number -**

```
DATA SEGMENT
        NUM DB ?
        FACT DB 1H
        RES DB 10 DUP ('$')
        AR DB   "ENTER NUMBER : $"
        ME DB 10,13,"FACTORIAL    : $"
DATA ENDS

CODE SEGMENT
ASSUME DS:DATA,CS:CODE
        FILLY:
        MOV AX,DATA
        MOV DS,AX
        LEA DX,AR
        MOV AH,9
        INT 21H
        MOV AH,1
        INT 21H
        SUB AL,30H
        MOV NUM,AL
        MOV AH,0
        MOV AL,FACT
        MOV CH,0
        MOV CL,NUM
        HA: MUL CL
        LOOP HA
        LEA SI,RES
        CALL AA
        LEA DX,ME
        MOV AH,9
        INT 21H
        LEA DX,RES
        MOV AH,9
        INT 21H
        MOV AH,4CH
        INT 21H
CODE ENDS
        AA PROC NEAR
        MOV CX,0
        MOV BX,10
        ED: MOV DX,0
        DIV BX
        ADD DL,30H
        PUSH DX
        INC CX
```

```
        CMP AX,9
        JG ED
        ADD AL,30H
        MOV [SI],AL
        AD: POP AX
        INC SI
        MOV [SI],AL
        LOOP AD
        RET
        AA ENDP
END FILLY
```

➔ **Output (Factorial) -**

5. **What is an interrupt to the processor? Explain IVT.**

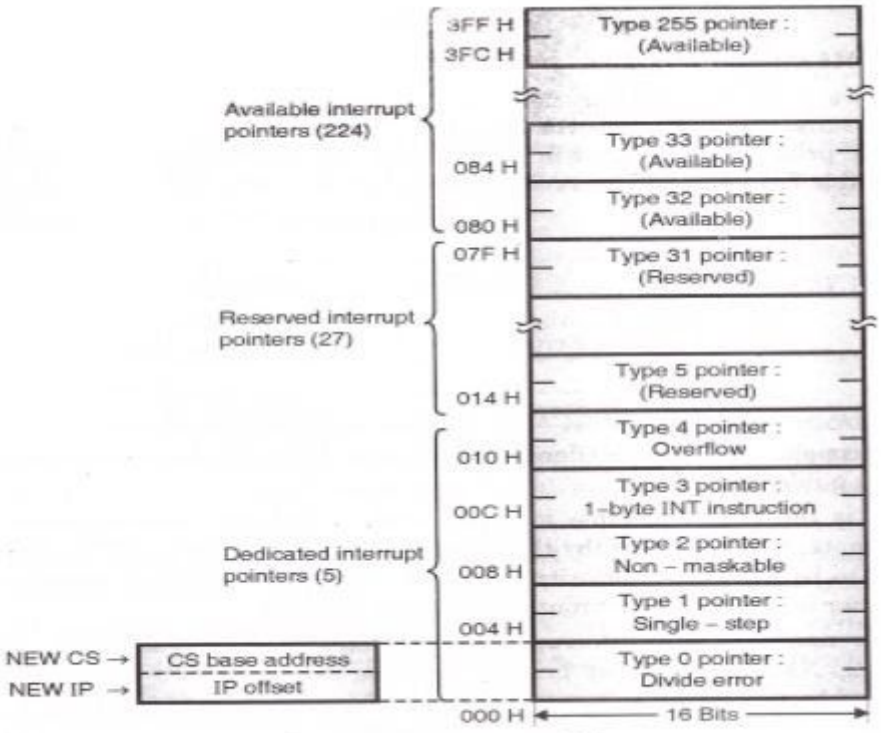
Ans:

An interrupt is a special condition that arises during the working of a microprocessor. The microprocessor services it by executing a subroutine called Interrupt Service Routine (ISR).

There are three sources of interrupts for 8086:

1. **Hardware interrupt-**
   These interrupts occur as signals on the external pins of the microprocessor. 8086 has two pins to accept hardware interrupts, NMI and INTR.
2. **Software interrupt-**
   These interrupts are caused by writing the software interrupt instruction INT n where 'n' can be any value from 0 to 255 (00H to FFH). Hence all 256 interrupts can be invoked by software.
3. **Error conditions (Exception or types)-**
   8086 is interrupted when some special conditions occur while executing certain instructions in the program. Example: An error in division automatically causes the INT 0 interrupt.

Interrupt Vector Table (IVT):



Interrupt vector table

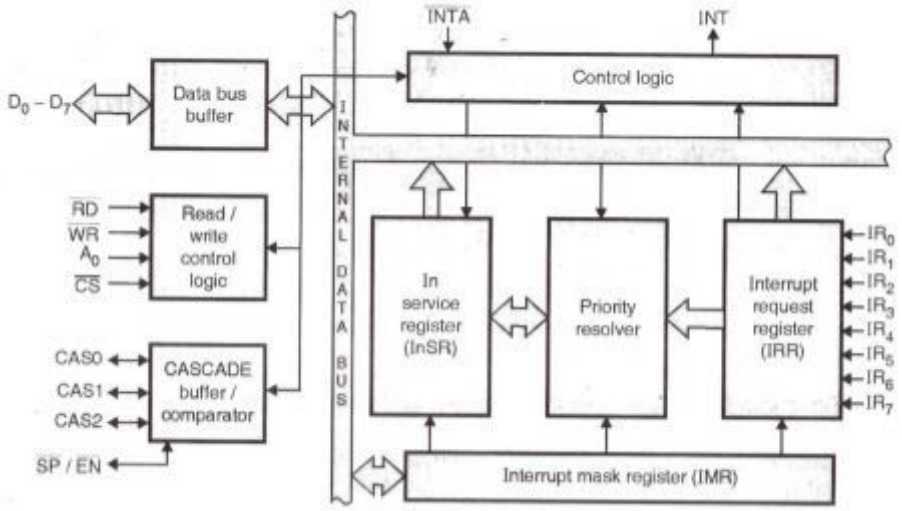
- ➔ The interrupt vector (or interrupt pointer) table is the link between an interrupt type code and the procedure that has been designated to service interrupts associated with that code. 8086 supports a total 256 types i.e. 00H to FFH.
- ➔ For each type, it has to reserve four bytes i.e. double words. This double word pointer contains the address of the procedure that is to service interrupts of that type.
- ➔ The higher addressed word of the pointer contains the base address of the segment containing the procedure. This base address of the segment is normally referred to as NEW CS.
- ➔ The lower addressed word contains the procedure's offset from the beginning of the segment. This offset is normally referred to as NEW IP.
- ➔ Thus NEW CS: NEW IP provides a NEW physical address from where user ISR routine will start.
- ➔ As for each type, four bytes (2 for NEW CS and 2 for NEW IP) are required; therefore the interrupt pointer table occupies up to the first 1k bytes (i.e. 256 x 4 = 1024 bytes) of low memory.
- ➔ The total interrupt vector table is divided into three groups namely,
  A. Dedicated interrupts (INT 0.....INT 4)
  B. Reserved interrupts (INT 5.....INT 31)
  C. Available interrupts (INT 32.....INT 225)

6. **Describe PIC block diagram with its operating mode.**

Ans:

8259 microprocessor is defined as a Programmable Interrupt Controller (PIC) microprocessor. There are 5 hardware interrupts and 2 hardware interrupts in 8085 and 8086 respectively. But by connecting 8259 with the CPU, we can increase the interrupt handling capability. 8259 combines the multi interrupt input sources into a single interrupt output. Interfacing of a single PIC provides 8 interrupts inputs from IR0-IR7.

The block diagram of 8259 is shown in the figure below:



Functional block diagram of 8259

The Block Diagram consists of 8 blocks which are – Data Bus Buffer, Read/Write Logic, Cascade Buffer Comparator, Control Logic, Priority Resolver and 3 registers- ISR, IRR, IMR.

➔ **Data bus buffer –**

This Block is used as a mediator between 8259 and 8085/8086 microprocessor by acting as a buffer. It takes the control word from the 8085 (let say) microprocessor and transfers it to the control logic of the 8259 microprocessor. Also, after the selection of Interrupt by 8259 microprocessor, it transfers the opcode of the selected Interrupt and address of the Interrupt service subroutine to the other connected microprocessor. The data bus buffer consists of 8 bits represented as D0-D7 in the block diagram. Thus, shows that a maximum of 8 bits data can be transferred at a time.

➔ **Read/Write logic –**
This block works only when the value of pin CS is low (as this pin is active low). This block is responsible for the flow of data depending upon the inputs of RD and WR. These two pins are active low pins used for reading and write operations.

➔ **Control logic –**
It is the centre of the microprocessor and controls the functioning of every block. It has pin INTR which is connected with other microprocessors for taking interrupt requests and pin INT for giving the output. If 8259 is enabled, and the other microprocessor Interrupt flag is high then this causes the value of the output INT pin high and in this way 8259 responds to the request made by other microprocessors.

➔ **Interrupt request register (IRR) –**
It stores all the interrupt levels which are requesting for Interrupt services.

➔ **Interrupt service register (ISR) –**
It stores the interrupt level which is currently being executed.

➔ **Interrupt mask register (IMR) –**
It stores the interrupt level which has to be masked by storing the masking bits of the interrupt level.

➔ **Priority resolver –**
It examines all the three registers and sets the priority of interrupts and according to the priority of the interrupts, interrupt with the highest priority is set in the ISR register. Also, it reset the interrupt level which has already been serviced in IRR.

➔ **Cascade buffer –**
To increase the Interrupt handling capability, we can further cascade more number of pins by using a cascade buffer. So, during increment of interrupt capability, CSA lines are used to control multiple interrupt structures.

Operating Modes of 8259 Microprocessor:
The various Operating Modes of 8259 Programmable Interrupt Controller are :

1. Fully Nested Mode of 8259,
2. Special Fully Nested Mode in 8259 (SFNM)
3. Rotating Priority Mode of 8259,
4. Special Mask Mode in 8259, and
5. Polled Mode in 8259.

1. Fully Nested Mode of 8259 (FNM) :

➔ After initialization, the 8259A operates in the fully nested mode so it is called default mode. The 8259 continues to operate in the Fully Nested Mode until the mode is changed through Operation Command Words. In this mode, IR0 has the highest priority and IR7 has the lowest priority. When the interrupt is acknowledged, it sets the corresponding bit in ISR. This bit will inhibit all interrupts of the same or lower level, however, it will accept higher priority interrupt requests. The vector address corresponding to this interrupt is then sent. The bit in the ISR will remain set until an EOI command is issued by the microprocessor at the end of the interrupt service routine. But if AEOI (Automatic End Of Interrupt) bit is set, the bit in the ISR resets at the trailing edge of the last INTA.

2. Special Fully Nested Mode in 8259 (SFNM):

➔ In the FNM, on the acknowledgement of an interrupt, further interrupts from the same level are disabled. Consider a large system which uses cascaded 8259s and where the interrupt levels within each slave have to be considered. An interrupt request input to a slave, in turn, causes the slave to place an interrupt request to the master on one of the master's inputs. Further interrupts to the slave will cause the slave to place requests to the master on the same input to the master, but these will not be recognised because further interrupts on the same input level are disabled by the master.

➔ The Special Fully Nested Mode (SFNM) is used to avoid this problem. The SFNM is set up by ICW4 during initialisation.

3. Rotating Priority Mode of 8259:

The Rotating Priority mode can be set in (i) Automatic Rotation, and (ii) Specific Rotation.

1. **Automatic Rotation:**

➔ In this mode, a device, after being serviced, receives the lowest priority. The device has just been serviced, will receive the seventh priority. Here IR3 has just been serviced.

IR_0	IR_1	IR_2	IR_3	IR_4	IR_5	IR_6	IR_7
4	5	6	7	0	1	2	3

2. **Specific Rotation:**

➔ In the Automatic Rotation mode, the interrupt request last serviced is assigned the lowest priority, whereas, in the Specific Rotation mode, the lowest priority can be assigned to an interrupt input (IR0 to IR7) thus fixes all other priorities.

➔ For example, if the lowest priority is assigned to IR2, other priorities are as shown below.

IR_0	IR_1	IR_2	IR_3	IR_4	IR_5	IR_6	IR_7
5	6	7	0	1	2	3	4

4. Special Mask Mode in 8259:

➔ If an interrupt is in service, then the corresponding bit is set in ISR and the lower priority interrupts are inhibited. Some applications may require an interrupt service routine to dynamically alter the system priority structure during its execution under software control. For example, the routine may wish to inhibit lower priority requests for a portion of its execution but enable some of them for another portion. In these cases, we have to go for a special mask mode. In the special mask mode, it inhibits further interrupts at that level and enables interrupts from all other levels (lower as well as higher) that are not masked. Thus any interrupt may be selectively enabled by loading the mask register.

5. Polled Mode in 8259:

➔ In this mode, the INT output is not used. The microprocessor checks the status of interrupt requests by issuing the poll command. The microprocessor reads contents of 8259A after issuing the poll command. During this read operation, the 8259A provides polled words and sets ISR bit of highest priority active interrupt request FORMAT.

I	X	X	X	X	W_2	W_1	W_0

I = 1 —> One or more interrupt requests activated.

I = 0 —> No interrupt request activated.

W2 W1 W0 —> Binary code of highest priority active interrupt request.