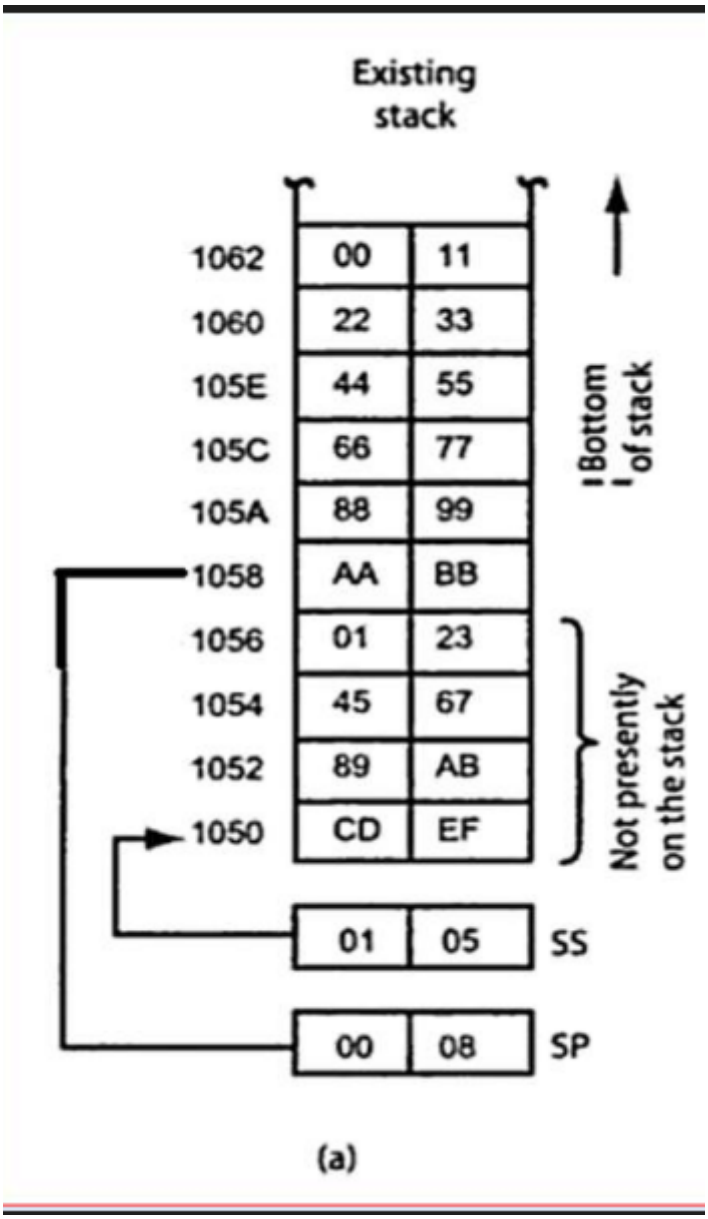
Microprocessors Module 3 Important Topics

Stack structure of 8086

- Stack is a sequence of RAM memory locations defined by a programmer
- Stack mechanism provides temporary storage of data
- Stack is Last in First out(LIFO)
- Stack is accessed using SP and SS register
- Each time a word is pushed into a stack
  - Value of SP is automatically decremented by 2
- Each Time a word is popped from the stack
  - Value of SP is automatically incremented by 2

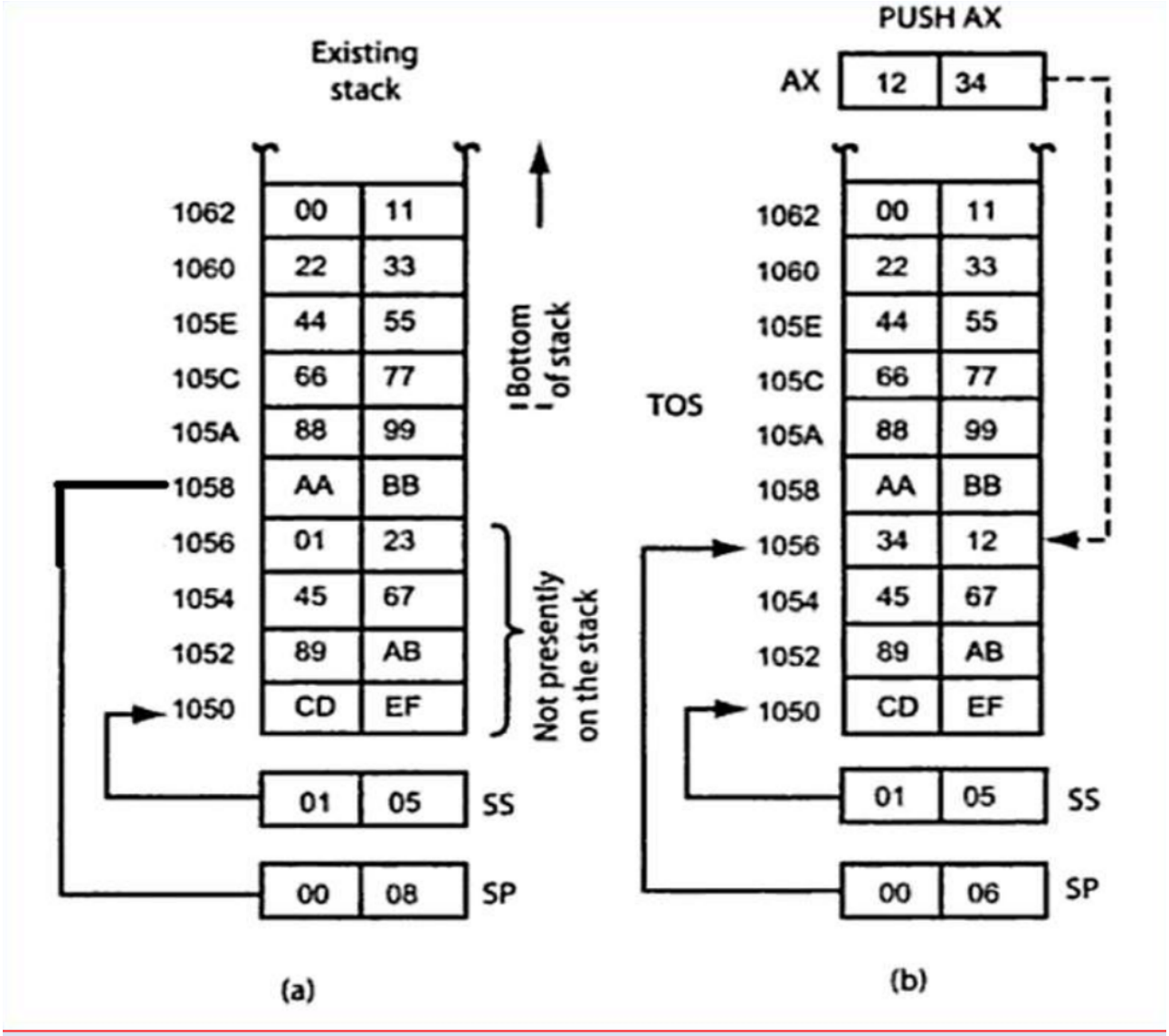


- We can find top and bottom of stack
  - Stack Segment (SS) register contains 105H
    - Bottom of the stack is derived from SS and Offset FFFEH
    - Bottom of stack address = SS + Offset = 1050H + FFFEH
      - = 1104EH
    - Top of stack is derived from SS and SP
    - Top of stack = SS +SP = 1050H + 0008H
      - = 1058H

Push Operation

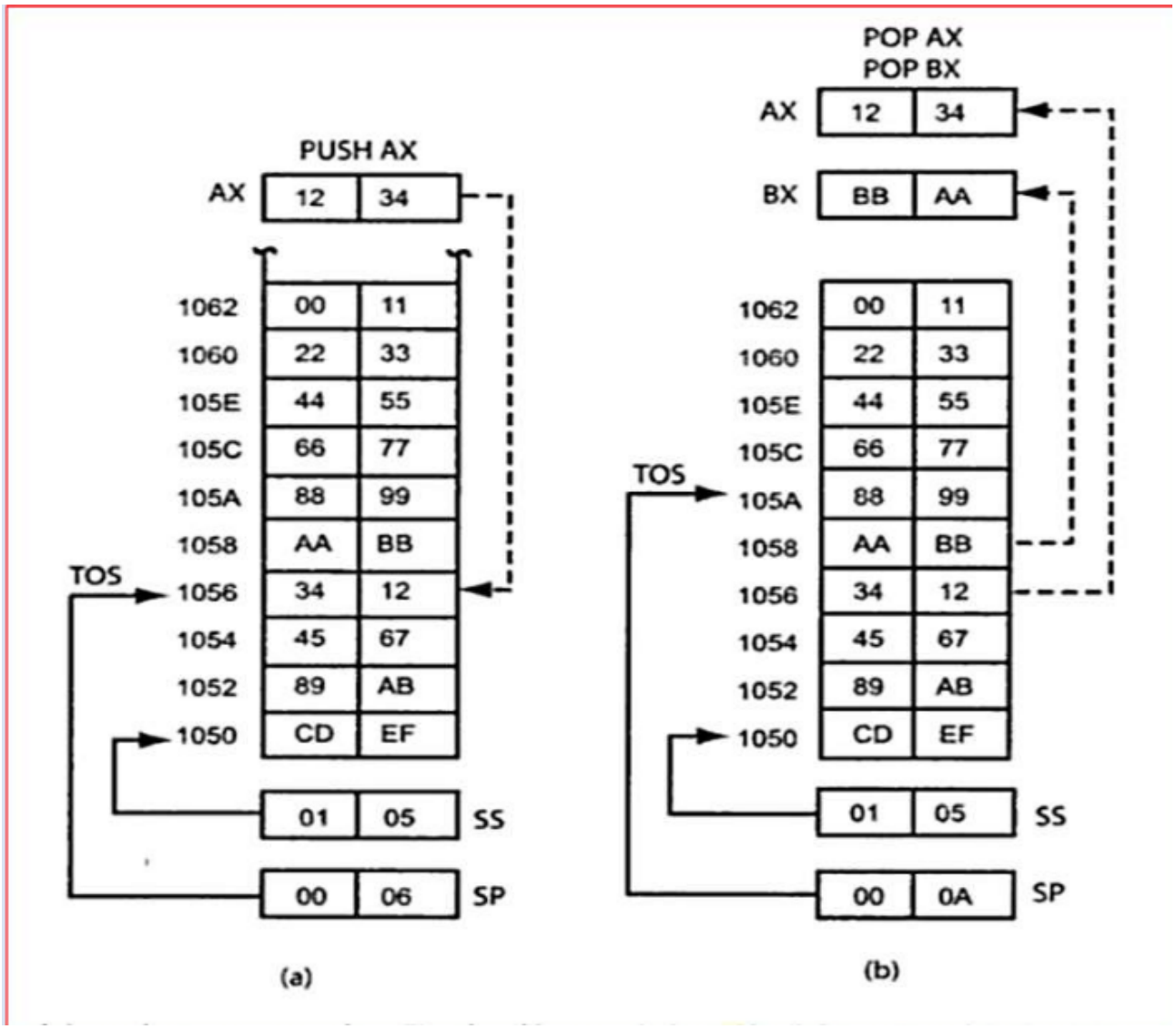
- Now we execute a push operation
  - We will be pushing AX
  - Here our top value is 1058 (Since the ones after that are unused)

        - The value after 1058 is 1056
    - So AX will be stored in in 1056
- AX has 2 values
    - LSB = 34, MSB = 12
    - LSB will be in 1056
    - MSB will be in 1057



- In the above figure you can see, when pushing, SP value is decremented from 08 to 06
- And the new pushed value occupied the unused space 1056

Pop Operation



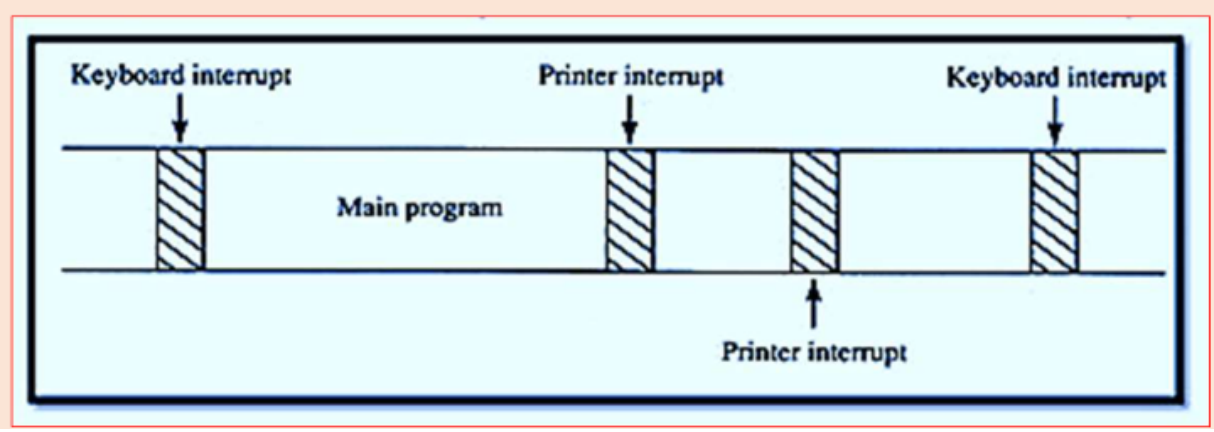
- We execute POP AX
  - Element at 1056 is popped off
  - This causes 8086 to read the value from the top of the stack (Our current top stack is at 1056) and put it into AX register as 1234H
  - As its a pop operation we will increment SP by 2
  - This makes the SP value go from 0006H -> 0008H
- We execute POP BX
  - Element at 1058 is popped off
  - SP is again incremented by 2
    - This makes the SP value from 0008H -> 000AH
  - The new top value is 105A
    - This is because Position at 1058 and 1056 are popped off

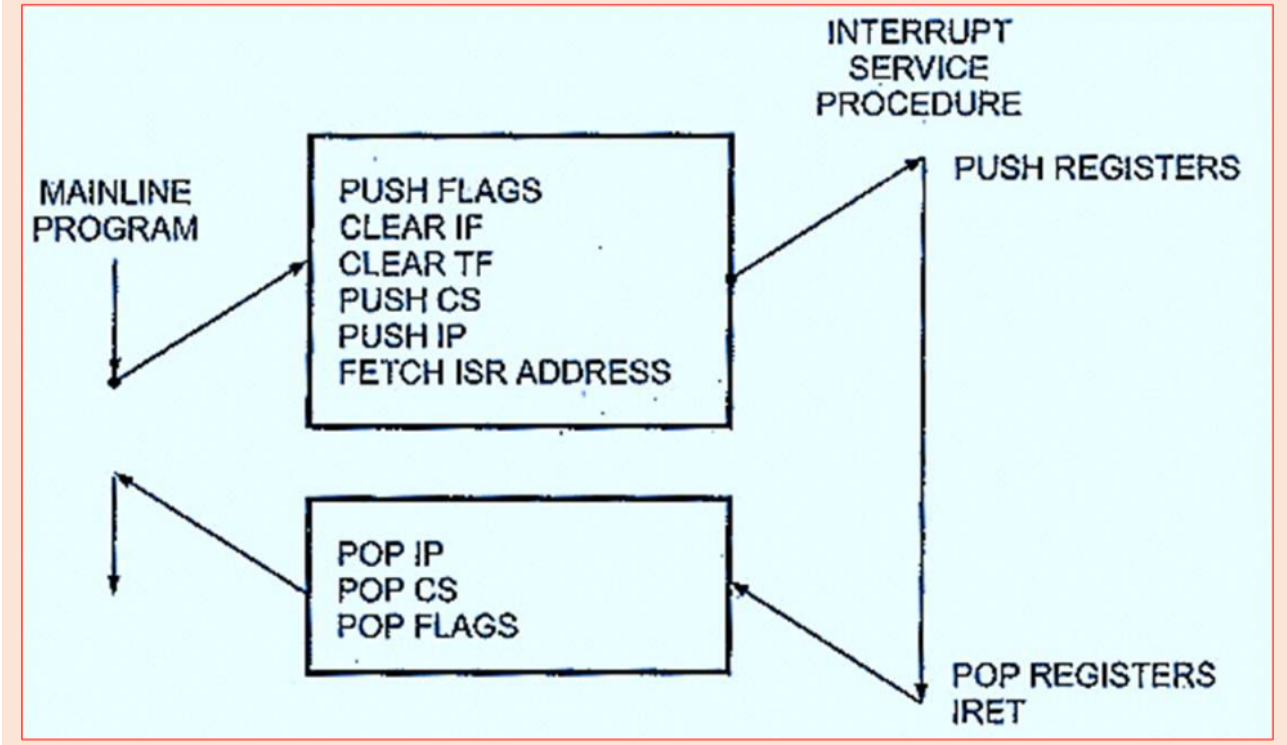
Interrupts and types

- Interrupt means to break the sequence of operation

- When CPU is executing a program, an interrupt does the following
  - breaks the normal sequence of execution of instructions
  - Divert its execution to a program called **Interrupt Service Routine** (ISR)
  - After executing ISR, The control is transferred back to the main program

Multiple interrupt processing capability

- When a number of devices interrupt the CPU at a time, if the processor is able to handle them properly, its said to have multiple interrupt processing capability
- The below figure we can see multiple interrupts
- 

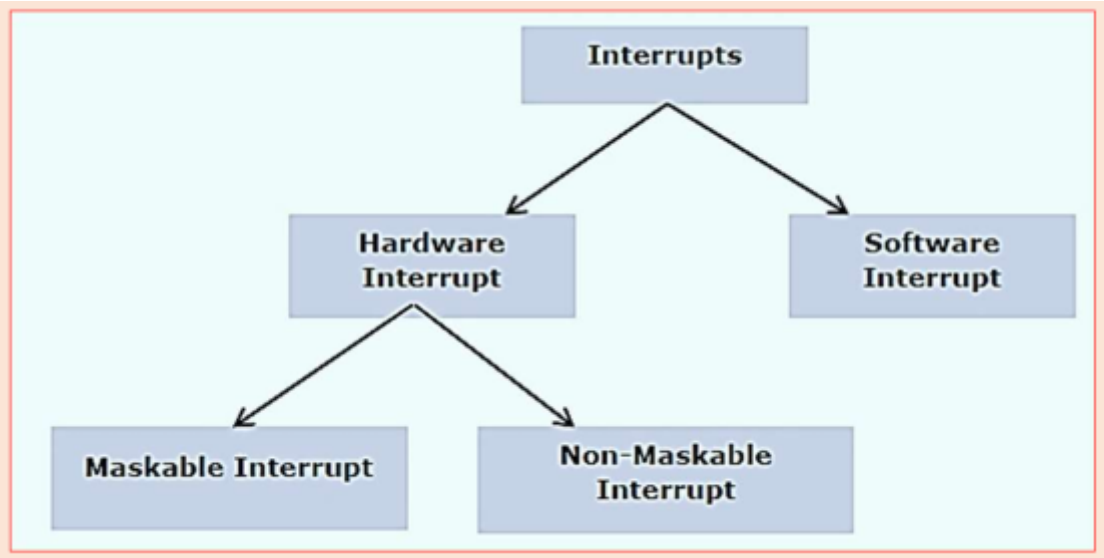
- Now lets see how these are being handled
- 

- In the above figure
  - Main Program gets executed
  - It has an interrupt and does the following
    - PUSH FLAGS

        - Stack pointer decremented by 2
        - Flag register values are Pushed
    - CLEAR IF
        - Clears interrupt flag
        - Disables the interrupt
    - CLEAR TF
        - Clears the trap flag
        - Which means to reset the trap flag
    - PUSH CS
        - Stack pointer decremented by 2
        - Contents of Code segment pushed to the stack
    - PUSH IP
        - Stack pointer decremented by 2
        - Contents of Instruction pointer pushed to stack
    - FETCH ISR ADDRESS
    - Go to Interrupt service procedure
    - PUSH REGISTERS
    - POP REGISTERS IRET
        - IRET means Interrupt Return
        - This instruction is used to return to the main program
            - POP IP
            - POP CS
            - POP FLAGS
    - After that we return to the program

Types of interrupts

- In 8086, there are 2 interrupt pins
    - NMI
    - INTR
- NMI -> Non Mistakable interrupt input pin
    - Any interrupt request at NMI input cannot be masked or disabled by any means
- INTR is a maskable interrupt
    - Can be masked using Interrupt Flag



Hardware interrupts

- Hardware interrupts are caused by peripheral devices by sending a signal through the specified pin in the microprocessor

NMI (Non Maskable interrupt)

- It has higher priority than INTR and is a type 2 interrupt
  When this interrupt is activated, these actions take place
- Completes the current instruction in progress
- Pushes Flag register values to stack
  - This step involves saving the contents of the flag register onto the stack.
  - The flag register contains various status flags that reflect the outcome of previous instructions or operations.
  - Saving these flags is crucial because the interrupt handler may modify the flags during its execution.
  - By pushing the flag register onto the stack, the processor preserves the original state of the flags.
- Pushes CS and IP value of return address to the stack
  - When an NMI occurs, the CPU needs to temporarily pause its current execution to handle the interrupt.
  - During this process, the CS (Code Segment) and IP (Instruction Pointer) values, which together specify the address of the next instruction to be executed, are saved onto the stack.
  - This saved CS:IP pair represents the return address, indicating where the CPU should resume normal program execution once the NMI handling is complete.
- IP is loaded from 00008H

- CS is loaded from 0000AH
- Interrupt Flag and trap flag are reset to 0

INTR(Maskable Interrupt)

- INTR is a maskable interrupt
- INTR Interrupt is activated by an IO Port
- When this interrupt is activated, the following takes place
  - First completes the current instruction
  - Activates INTA (Interrupt Acknowledge) Output and receives Interrupt Type, Say X
    - This INTA signal is sent to the external devices connected to the INTR line, indicating that the microprocessor has recognized and acknowledged the interrupt request.
  - Flag Register Value, CS Value and IP value of return address are pushed to stack
  - IP value is loaded from contents of next word location X x 4
    - So IP Values = content of X x 4 where X is the next word location
  - CS is loaded from contents of the next word location
  - Interrupt flag and trap flag are set to 0

Software interrupts

- These are instructions that are inserted within the program to generate interrupts
- There are 256 software interrupts in 8086 microprocessor
- These instructions are of the following format
  - INT type
    - Here type can be Ranging from 00 to FF
- Some important software interrupts are
  - TYPE 0
    - corresponds to division by zero(0)
  - TYPE 1
    - used for single step execution for debugging program
  - TYPE 2
    - Represents NMI and used in power failure conditions
  - TYPE 3
    - Represents break point interrupt
  - TYPE 4
    - Overflow Interrupt

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

Interrupt Vectors

- The starting address of an ISP(Interrupt Service Provider) is called the interrupt vector. There fore the table is called interrupt vector table.
- The interrupt vector table contains 256 four byte entries
    - Containing CS:IP
- It contains the Address of Interrupt service provider (ISR)

8259 Programmable Interrupt Controller (PIC)

- Programmable interrupt controllers are used to enhance the number of interrupts of a microprocessor
- It is also known as a priority interrupt controller and was designed by Intel to increase the interrupt handling ability of the microprocessor
- An 8259 PIC never services an interrupt; it simply forwards the interrupt to the processor for the execution of interrupt service routine.

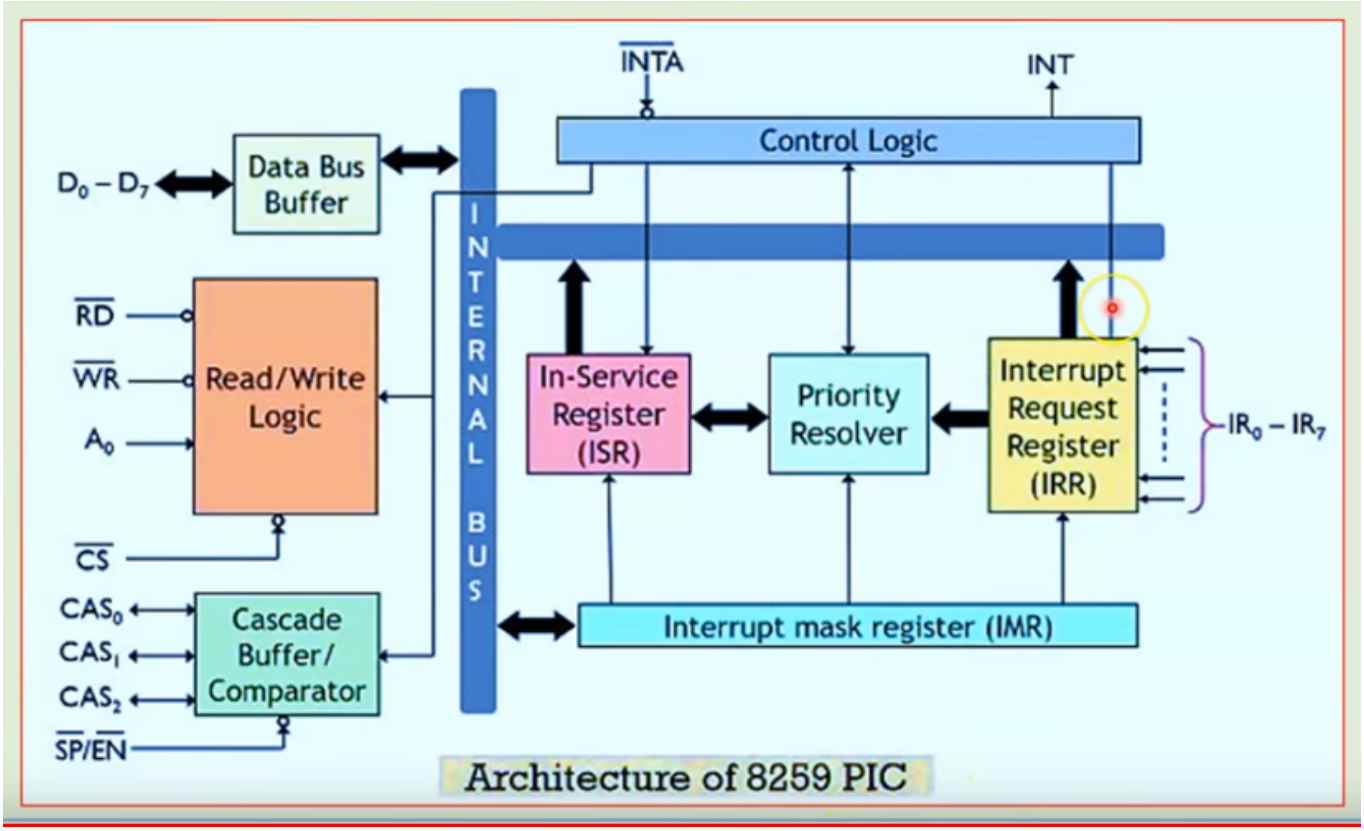
Need of Programmable Interrupt Controller

- Vectored Interrupts are those interrupts who ISR address is known to the process
    - In the case of vectored interrupts, the processor holds the ISR address
- In Non Vectored interrupts, The interrupt generating device provides the ISR address to microprocessor
- Why 8259 is used?
    - In 8086/8085 there are 5 major interrupts
    - Which has fixed number of lines present in the chip
    - But there are many devices connected to a processor
        - So in this case, the processor must have more number of lines
            - So it can handle several interrupts
    - But its not practically possible to increase number of lines
    - To overcome this, 8259 is used
        - It allows combining of multiple interrupts
            - Providing them to processor based on priority

- 8259 is used to combine non vectored interrupts
- Basically the external devices initially interrupt the 8259 and further the 8259 interrupts the microprocessor.

Features of 8259

- The 8259 programmable interrupt controller has 8 interrupt pins and can handle 8 interrupt inputs.
- The priority of interrupts in 8259 can be programmed.
- A single 8259 can handle 8 interrupt inputs but by cascading multiple 8259, it can handle maximal 64 interrupt inputs
  - Here cascading means placing them side by side, So suppose we have 8 8259s
  - Then the total inputs are 8 x 8 = 64
- 8259 can handle either edge-triggered or level-triggered interrupt request at a time.
- 8259 allows individual masking of each generated interrupt using interrupt mask register.
  - "masking" refers to the ability to selectively enable or disable specific interrupt inputs.
- If multiple interrupts are generated, then 8259 holds the status of Interrupts that are masked, in-service and pending.
- it reduces the software and real-time overhead generated due to handling multilevel priority interrupts



- The above figure is the architecture of 8259 PIC

- D0 to D7
  - Data lines
- RD, WR, A0
  - Read, Write and Address bit 0
- CS (Chip select)
  - CS is an active-low signal, meaning it is considered "selected" when it is low (0).
  - When the microprocessor wants to communicate with the 8259, it asserts the CS signal by pulling it low.
- **CAS0, CAS1, CAS2**
  - Represents the cascade lines
- IRR (Interrupt Request Service)
  - Represents the current state of interrupt requests. It holds information about which interrupt lines are currently requesting service.
  - IR0 to IR7
    - Represents the 8 input lines of interrupt
- Interrupt Mask Register
  - Its primary function is to enable or disable specific interrupt lines.
  - Each bit in the Interrupt Mask Register corresponds to a specific interrupt line, and the state of the bit determines whether the corresponding interrupt line is masked (disabled) or unmasked (enabled).