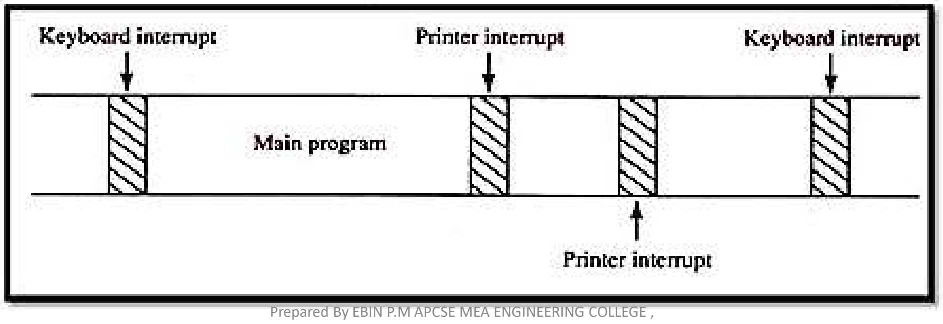
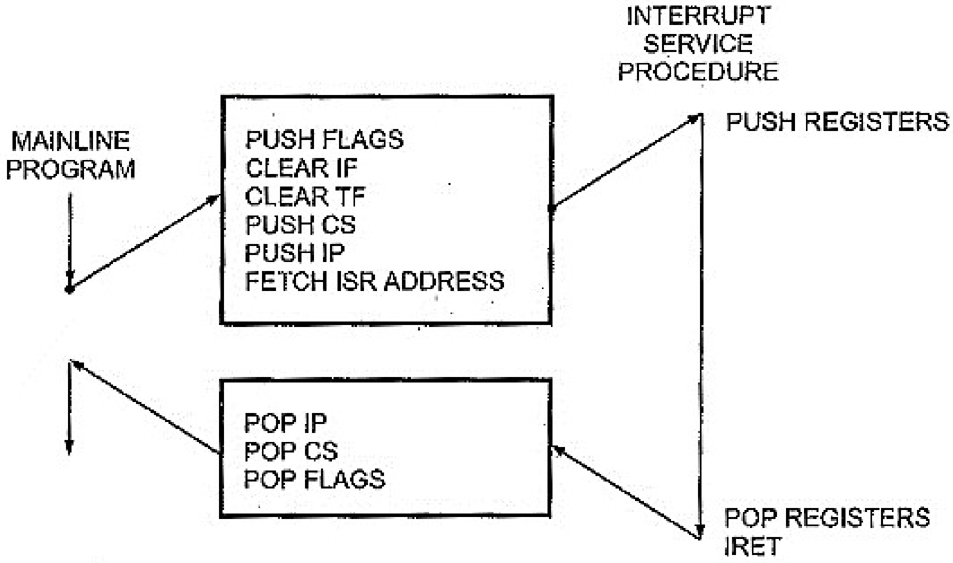
MODULE 3 : INTERRUPTS IN 8086

- **Interrupt** – break the sequence of operation
- While the CPU executing a program, an interrupt breaks the normal sequence of execution of instructions, divert its execution to some other program called **Interrupt Service Routine (ISR)**
- After executing ISR, the control is transferred back again to the main program
- Whenever a number of devices interrupt the CPU at a time, and if the processor is able to handle them properly, it is said to have **multiple interrupt processing capability**.

- The following time line shows typing on a keyboard, a printer removing data from memory, and a program executing.
- The keyboard interrupt service procedure, called by the keyboard interrupt, and the printer interrupt service procedure called by the printer interrupt.
- each take little time to execute





INTERRUPT SERVICE PROCEDURE

MAINLINE PROGRAM

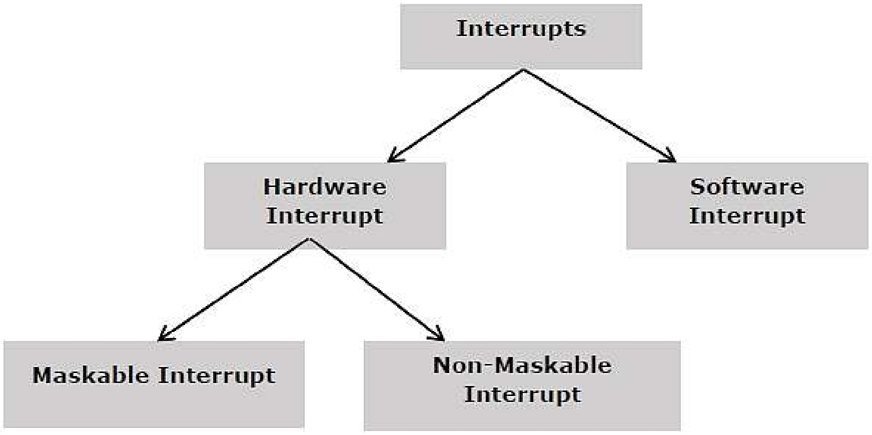
PUSH FLAGS
CLEAR IF
CLEAR TF
PUSH CS
PUSH IP
FETCH ISR ADDRESS

PUSH REGISTERS

POP IP
POP CS
POP FLAGS

POP REGISTERS
IRET

Types of interrupts we have in a 8086 microprocessor



Interrupts

Hardware Interrupt

Software Interrupt

Maskable Interrupt

Non-Maskable Interrupt

- In 8086, there are two interrupt pins. **NMI** and **INTR**

- NMI is a **nonmaskable interrupt** input pin which means that any interrupt request at NMI input cannot be masked or disabled by any means.
- The INTR interrupt may be masked using the Interrupt Flag (IF)

Hardware Interrupts

- Hardware interrupts are those interrupts which are caused by any peripheral device by sending a signal through a specified pin to the microprocessor.
- 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. One more interrupt pin associated is INTA called interrupt acknowledge.

a) NMI (Non Maskable Interrupt)

- It is a single nonmaskable interrupt pin (NMI) having higher priority than the maskable interrupt request pin (INTR)and it is of type 2 interrupt.

When this interrupt is activated, these actions take place –

1. Completes the current instruction that is in progress.

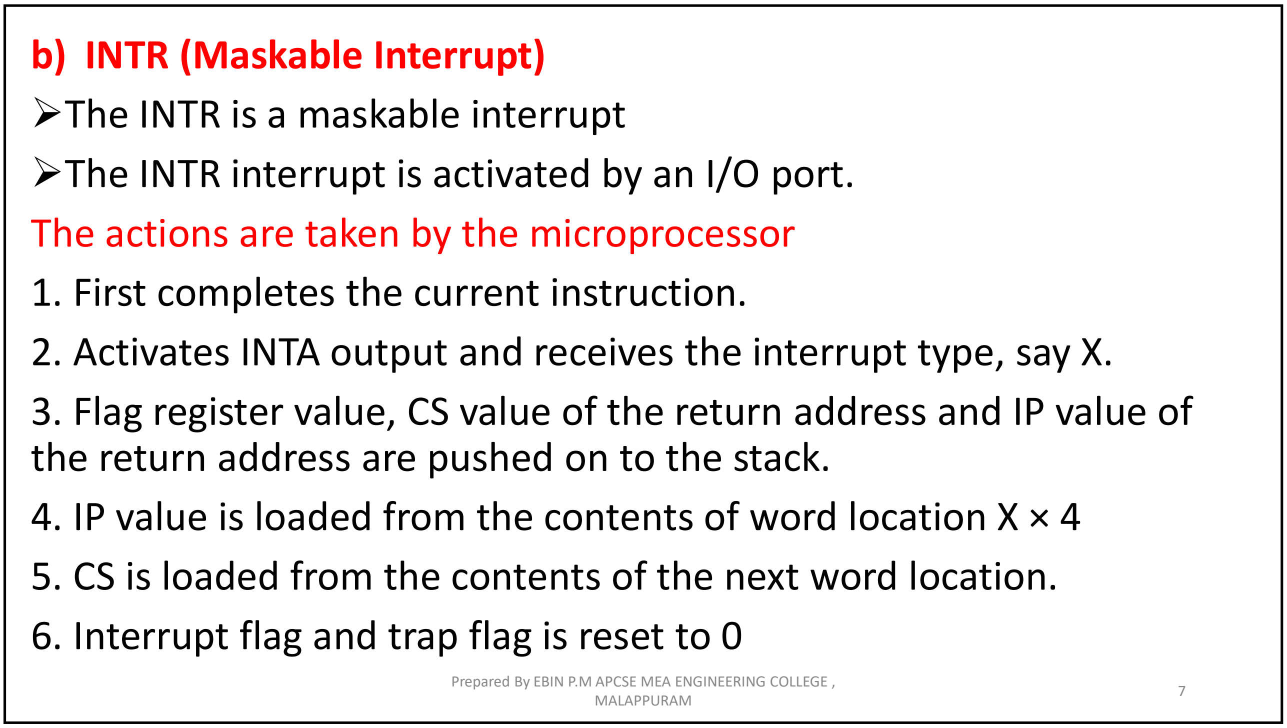
2. Pushes the Flag register values on to the stack.

3. Pushes the CS (code segment) value and IP (instruction pointer) value of the return address on to the stack.

4. IP is loaded from the word location 00008H.

5. CS is loaded from the word location 0000AH.

6. Interrupt flag and trap flag are reset to 0.

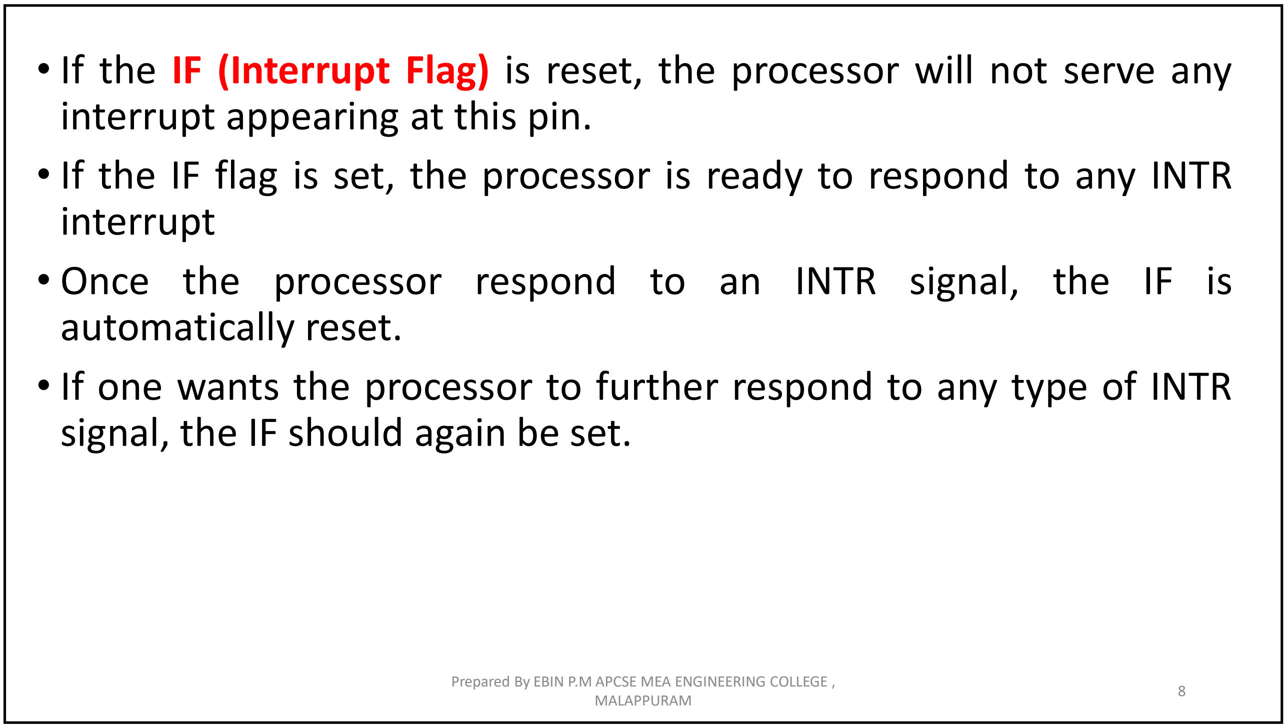


b) INTR (Maskable Interrupt)

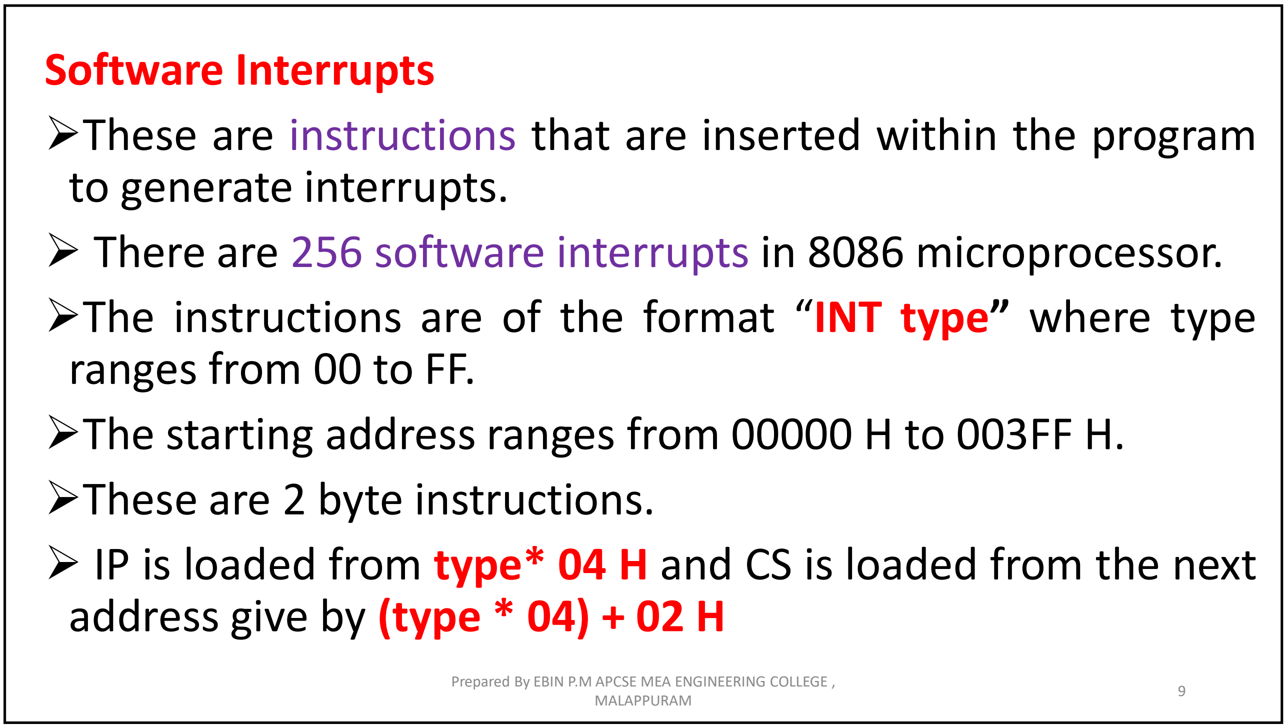
- ➢The INTR is a maskable interrupt
- ➢The INTR interrupt is activated by an I/O port.

The actions are taken by the microprocessor

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

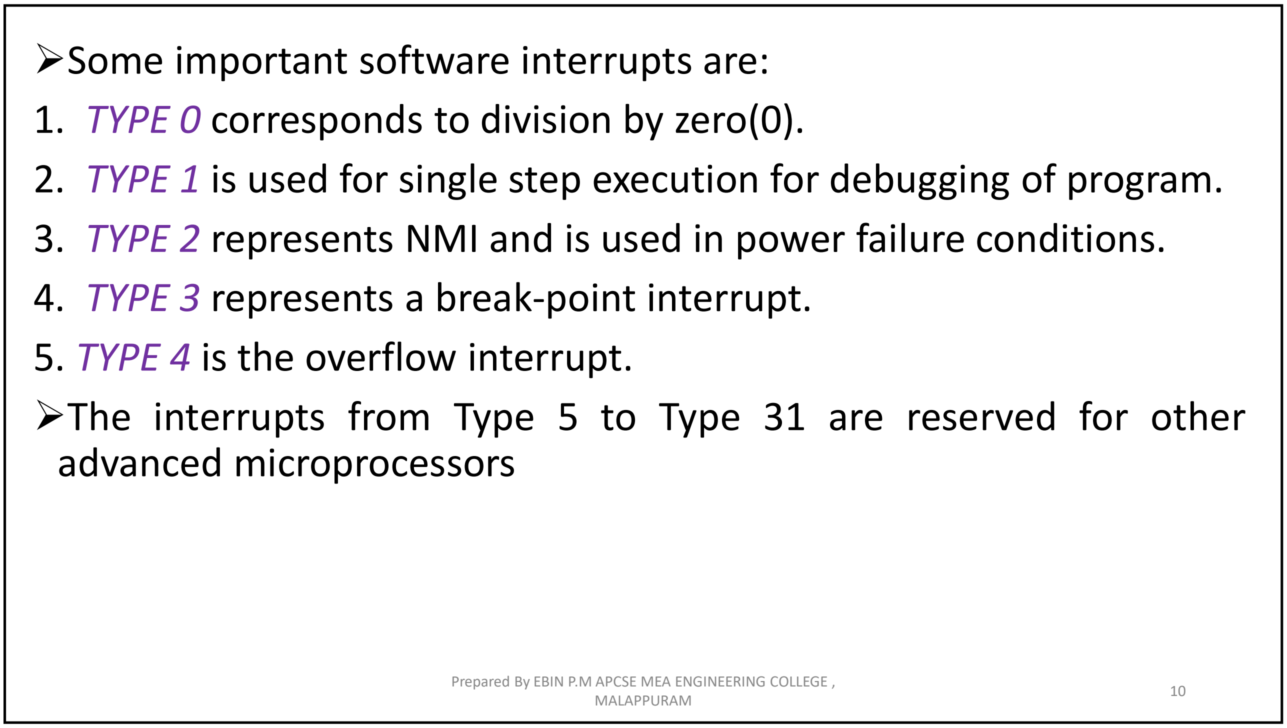


- If the **IF (Interrupt Flag)** is reset, the processor will not serve any interrupt appearing at this pin.
- If the IF flag is set, the processor is ready to respond to any INTR interrupt
- Once the processor respond to an INTR signal, the IF is automatically reset.
- If one wants the processor to further respond to any type of INTR signal, the IF should again be set.



Software Interrupts

- These are instructions that are inserted within the program to generate interrupts.
- There are 256 software interrupts in 8086 microprocessor.
- The instructions are of the format “**INT type**” where type ranges from 00 to FF.
- The starting address ranges from 00000 H to 003FF H.
- These are 2 byte instructions.
- IP is loaded from **type* 04 H** and CS is loaded from the next address give by **(type * 04) + 02 H**



- Some important software interrupts are:

1. *TYPE 0* corresponds to division by zero(0).
2. *TYPE 1* is used for single step execution for debugging of program.
3. *TYPE 2* represents NMI and is used in power failure conditions.
4. *TYPE 3* represents a break-point interrupt.
5. *TYPE 4* is the overflow interrupt.

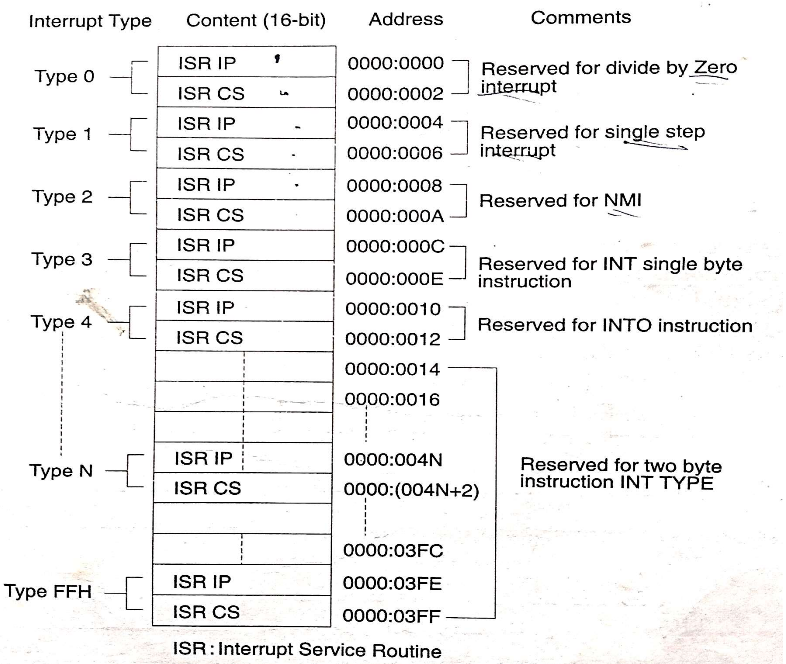
- The interrupts from Type 5 to Type 31 are reserved for other advanced microprocessors

Interrupt Service Routine (ISR)

- 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 the CS: IP
- Interrupt vector table is located in the first 1024 bytes of memory at addresses 000000H – 0003FFH . It contains the address of the interrupt service provider



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	
		0000:0014	Reserved for two byte instruction INT TYPE
		0000:0016	
Type N	ISR IP	0000:004N	
	ISR CS	0000:(004N+2)	
		0000:03FC	
Type FFH	ISR IP	0000:03FE	
	ISR CS	0000:03FF	

ISR : Interrupt Service Routine

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

INTERRUPT & POLLING

- ➢The advantage of interrupts is that the microprocessor can serve many devices (not all at the same time, of course); each device can get the attention of the microprocessor based on the priority assigned to it.
- ➢The polling method cannot assign priority because it checks all devices in a round-robin fashion.
- ➢More importantly, in the interrupt method the microprocessor can also ignore (mask) a device request for service.
- ➢This is not possible with the polling method.
- ➢polling method wastes much of the microprocessor’s time by polling devices that do not need service.
- ➢So interrupts are used to avoid tying down the microprocessor.