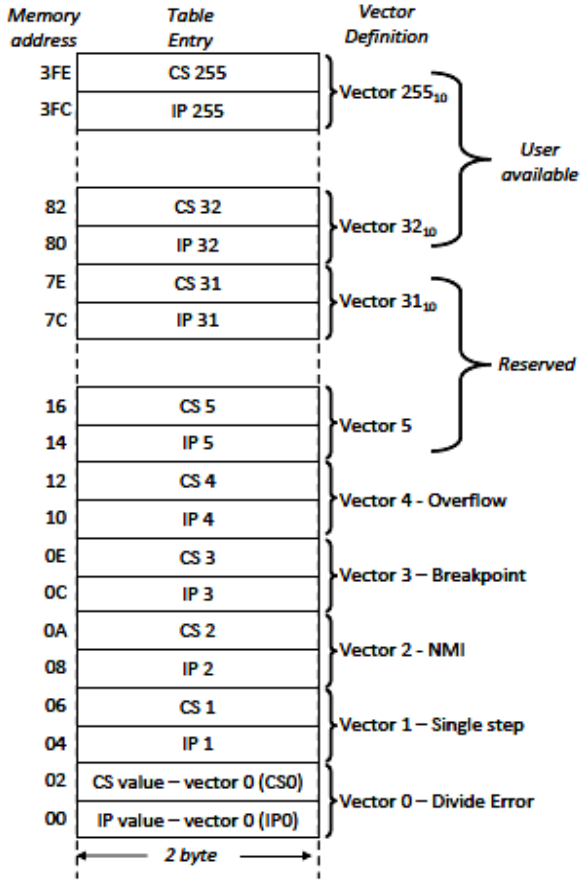
Interrupt is the method of creating a ***temporary halt*** during program execution and allows peripheral devices to access the microprocessor. The microprocessor responds to that interrupt with an **ISR** (Interrupt Service Routine) or **Interrupt Handler**.

ISR (Interrupt Service Routine): is a short program to instruct the microprocessor on how to handle the interrupt. **For example**, when the printer causes an interrupt, the printer **Driver** (Interrupt Service Routine) is called, *but how does the microprocessor know* ***where to jump****?*

The **8086** microprocessor is capable of implementing any combination of up to **256 interrupts**.

An ***Interrupt Vector Table (IVT)*** is stored in the first **1 kbyte** of memory (starting at address 00000 H and ending at 003FF H). ***IVT*** is a pointer table to indicate the location of service routines corresponding to interrupt **types 0 to 255**.



Interrupt vector table of the 8086 microprocessor

- ✓ The ***CS and IP*** in the interrupt vector table indicate the location of the service routine for the corresponding interrupt.

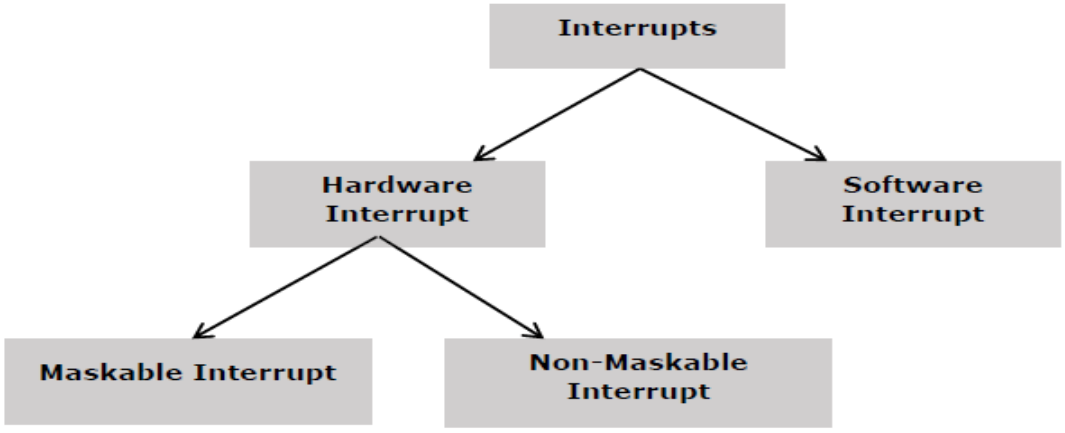
- ✓ For example, when an interrupt of **type 32** occurs, the microprocessor looks at address 32 * 4 = 128 = **80 H** and (after pushing the current CS and IP) loads the new values of CS and IP from 80h to jump to the interrupt service routine.

After the ISR execution, control returns to the main routine where it was interrupted. ***In the 8086 microprocessor following tasks are performed when the microprocessor encounters an interrupt:***

1. The value of the flag register is pushed into the stack. It means that first, the value of SP (Stack Pointer) is decremented by two then the value of the flag register is pushed to the memory address of the stack segment.
2. The value of the starting memory address of CS (Code Segment) is pushed into the stack.
3. The value of IP (Instruction Pointer) is pushed into the stack.
4. IP is loaded from word location (Interrupt type) * 04.
5. CS is loaded from the following word location.
6. Interrupt flag, and Trap flag are reset to 0.

❖ Types of Interrupts

The different types of interrupts present in the 8086 microprocessor are given by the following figure:



❖ Hardware Interrupts

Hardware interrupt is 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**. One more interrupt pin associated is **INTA** called interrupt acknowledge.

1- NMI (Non-Maskable Interrupt)

It is a single non-maskable interrupt pin (NMI) having higher priority than the maskable interrupt request pin (INTR), NMI cannot be disabled. After its execution, this interrupt generates a **TYPE 2** interrupt. IP is loaded from the word location 00008 H, and CS is loaded from the word location 0000A H. When this interrupt is activated, these actions take place :

- Store the status of the current instruction that is in progress.
- Pushes the Flag register values onto the stack.
- Pushes the CS (code segment) value and IP (instruction pointer) value of the return address onto the stack.
- IP is loaded from the contents of the word location 00008H.
- CS is loaded from the contents of the next word location 0000AH.
- Interrupt flag and trap flag are reset to 0.

2- INTR (Maskable Interrupt)

The INTR is a maskable interrupt because the microprocessor will be interrupted only if interrupts are enabled using **Set interrupt Flag instruction (*STI*)**. It should not be enabled using **Clear Interrupt Flag instruction(*CLI*)**.

The INTR interrupt is activated by ***an I/O port***. If the interrupt is enabled and NMI is disabled, then the microprocessor first completes the current execution and sends ***'0' on INTA pin twice***. The first '0' means INTA informs the external device to get ready and during the second '0' the microprocessor receives the 8 bit, say X, from the ***Programmable Interrupt Controller***.

programmable Interrupt Controller: The 8259 a programmable interrupt controller chip accepts interrupts from up to eight different devices, If any one of the

devices requests service, the 8259 will toggle an interrupt output line (connected to the CPU) and pass a programmable interrupt vector to the CPU.

When the Maskable-Interrupt is activated, these actions are taken by the 8086 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.

Maskable Interrupt	Non-Maskable Interrupt
Maskable interrupt is a hardware Interrupt that can be disabled or ignored by the instructions of CPU.	A non-maskable interrupt is a hardware interrupt that cannot be disabled or ignored by the instructions of CPU.
When maskable interrupt occur, it can be handled after executing the current instruction.	When non-maskable interrupts occur, the current instructions and status are stored in stack for the CPU to handle the interrupt.
Maskable interrupts help to handle lower priority tasks, such as interface with peripheral device.	Non-maskable interrupt help to handle higher priority tasks such as power failure, smoke detector.
Operation can be masked or made pending.	Operation Cannot be masked or made pending.
It's a Level-Triggered Interrupt	It's an Edge-Triggered Interrupt

❖ Software Interrupts (INT- Interrupt instruction with Type Number)

These instructions are inserted at the desired position into the program to create interrupts. In the 8086 microprocessor.

These interrupt instructions can be used to test the working of various interrupt handlers. The instructions are of the format ***INT Type,*** it is 2-byte instruction. First byte provides the ***op-code*** and the second byte provides the ***Interrupt Type Number***.

IP is loaded from (**type * 04 H)**, and CS is loaded from the following address given by **(type * 04) + 02 H**.

The starting address for **type 0** interrupt is 000000H, for **type1** interrupt is 00004H, similarly for **type 2** is 00008H and so on. Some important software interrupts are:

- **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.

The interrupts from ***Type 5 to Type 31*** are reserved for **other advanced microprocessors**, and interrupts from ***32 to Type 255*** are available for **hardware and other software interrupts**.

INT 3-Break Point Interrupt Instructions

These instructions are inserted into the program so that when the processor reaches there, then it stops the normal execution of program and follows the break-point procedure.

Its execution includes the following steps:

- Flag register value is pushed on to the stack.
- CS value of the return address and IP value of the return address are pushed on to the stack.
- IP is loaded from the contents of the word location 3×4 = 0000CH

- CS is loaded from the contents of the next word location.
- Interrupt Flag and Trap Flag are reset to 0

INTO - Interrupt on Overflow Instruction

As the name suggests it is a conditional interrupt instruction, i.e. it is active only when the overflow flag is set to 1 and branches to the interrupt handler whose interrupt type number is ***4***. If the overflow flag is reset then, the execution continues to the next instruction.

Its execution includes the following steps:

- Flag register values are pushed on to the stack.
- CS value of the return address and IP value of the return address are pushed on to the stack.
- IP is loaded from the contents of word location 4×4 = 00010H
- CS is loaded from the contents of the next word location.
- Interrupt flag and Trap flag are reset to 0

HW: At what address are CS_{50} and IP_{50} stored in memory **??**