CHAPTER 5: INTERRUPTS

INTRODUCTION

Interrupt is considered as an emergency signal to which the MP responds as soon as possible. When the microprocessor receives an interrupt signal, it suspends the current executing program and jumps to an interrupt service routine (ISR) to respond to the incoming interrupt When a device interrupts, it actually wants the microprocessor to give a service which is equivalent to asking the microprocessor to call a subroutine. This subroutine is called Interrupt Service Routine (ISR).

SOURCES OF INTERRUPTS

There are three sources of interrupts and they are as follows:

1. Processor Interrupt
2. Software Interrupt
3. Hardware Interrupt

Processor Interrupt

These interrupts are generated by processor itself, usually in response to an error condition. For example: In 8086 Type 0 interrupt occurs when attempt to divide by zero which is a processor interrupt.

Software Interrupt

These are special instructions that trigger an interrupt response to processor.
In 8086 the general form of software interrupt instruction is INT nnH (eg: INT 21H)

Hardware Interrupt

Hardware interrupts are interrupt request initiated by external hardware.
8086 have two pins reserved for hardware interrupts. They are NMI and INTR

CLASSIFICATIONS OF INTERRUPTS

Interrupts can be classified as:

- Maskable Interrupt or Non-Maskable Interrupt
- Vectored Interrupt or Non-Vectored Interrupt

Maskable Interrupt

- The interrupts which can be blocked or delayed by using instructions are called maskable interrupts.
- In 8085, the RESET interrupts (RST 5.5, RST 6.5 and RST 7.5) and INTR are maskable interrupts. They can be enabled/disabled by using instructions EI/DI.
- In 8086, INTR is maskable interrupt. It can be enabled/disabled by using instructions STI/CLI.

- Those interrupts which cannot be blocked by instructions are termed as non-maskable interrupts.
- In 8085, TRAP is only non-maskable interrupt and it is used for power failure and emergency cutoff.
- In 8086, NMI is non-maskable interrupt.

Vectored Interrupt

- The interrupts for which address of ISR is already known to MP are called vectored interrupts.
- In 8085, RESET interrupts (RST 5.5, RST 6.5 and RST 7.5) are vectored interrupts.

Interrupt	**Vector Address (Hex)**
RST 5.5	002C
RST 6.5	0034
RST 7.5	003C

Non-Vectored Interrupt

- In non-vectored interrupts, the interrupting device needs to supply the address of the ISR to the microprocessor.
- In 8085, INTR is non-vectored interrupt.

8085 INTERRUPTS

8085 microprocessor consists of five interrupt signals: INTR, RST 5.5, RST 6.5, RST 7.5 and TRAP

1. INTR (Interrupt Request)

- The INTR is only non-vectored interrupt.
- INTR is maskable interrupt and can be masked by using EI (Enable Interrupt)/DI (Disable Interrupt) instruction pair.
- INTR can be used for external hardware interrupts for various applications.

2. RESET Interrupts

- 8085 consists of three reset interrupts: RST 5.5, RST 6.5 and RST 7.5
- They are vectored interrupts whose address are defined in interrupt vector table
- They are maskable interrupts and can be enabled or disabled individually by using RIM and SIM instructions.

Interrupt	**Vector Address (Hex)**
TRAP	0024
RST 5.5	002C
RST 6.5	0034
RST 7.5	003C

Fig: 8085 Interrupt Vector Table

3. TRAP

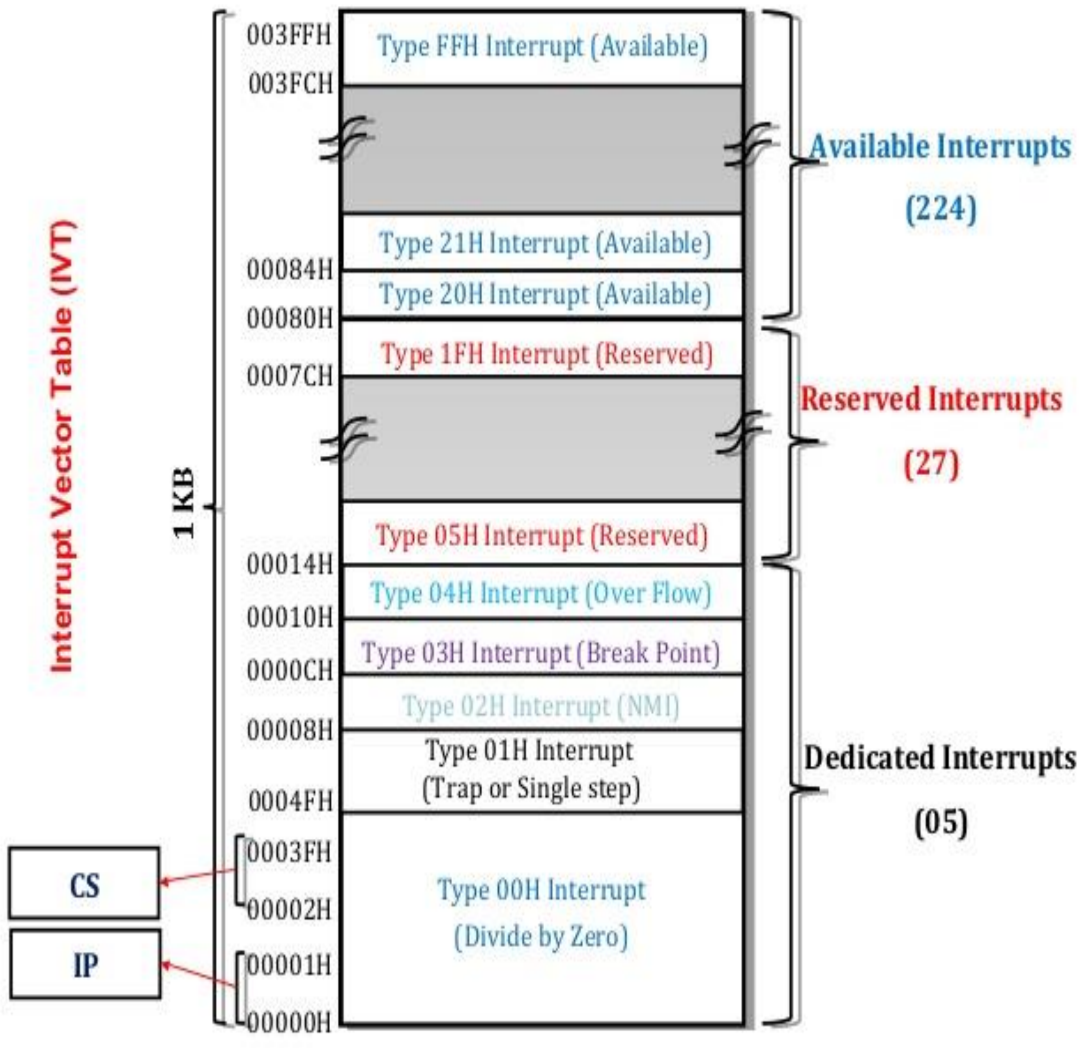
- It is only non-maskable interrupt of 8085 MP.
- It is a vectored interrupt whose address is defined in interrupt vector table.
- It is used as external hardware interrupt source and is used for power failure and emergency shutoff instruction.

8086 INTERRUPTS

- An 8086 interrupt can come from any one of 3 sources.
- One source is an external signal applied to the Nonmaskable Interrupt (NMI) or to the (INTR) input pin. An interrupt caused by a signal applied to one of these inputs (NMI or INTR) is referred as **Hardware Interrupt**.
- A second source of an interrupt is execution of the interrupt instruction INT nn. This is referred as Software **Interrupt.**
- The third source of an interrupt is some error conditions produced in 8086, by the execution of an instruction, referred as processor interrupt.

INTERRUPT VECTOR TABLE (IVT)

- 8086 consists of 256 interrupts stored from memory location 00000H to 003FFH i.e. 1024 Bytes (1KB) of memory.
- These 256 interrupts are stored in memory location forming a table which is called Interrupt Vector Table (IVT).
- An Interrupt Vector contains the address (segment CS and offset IP) of the Interrupt Service Routine.
- Each vector is a 4 byte long and contains the starting address of the Interrupt Service Routine.
- The first 2 bytes of the vector contain the offset address (IP) and the last two bytes contain the segment address (CS).



Interrupt Vector Table (IVT)
1 KB
003FFH
003FCH
Type FFH Interrupt (Available)
Available Interrupts
(224)
Type 21H Interrupt (Available)
00084H
Type 20H Interrupt (Available)
00080H
Type 1FH Interrupt (Reserved)
0007CH
Reserved Interrupts
(27)
Type 05H Interrupt (Reserved)
00014H
Type 04H Interrupt (Over Flow)
00010H
Type 03H Interrupt (Break Point)
0000CH
Type 02H Interrupt (NMI)
00008H
Type 01H Interrupt (Trap or Single step)
0004FH
Dedicated Interrupts
(05)
0003FH
00002H
Type 00H Interrupt (Divide by Zero)
00001H
00000H
CS
IP

8086 PREDEFINED INTERRUPT TYPES

1. **DIVIDE BY ZERO INTERRRUPT: TYPE 0**
- Divide error occurs when the result of division overflow or whenever an attempt is made to divide by zero.
- The 8086 type 0 is automatic and cannot be disabled in any way

2. **SINGLE STEP INTERRUPT : TYPE 1**
- If the 8086 trap flag is set, the 8086 will automatically do a type 1 interrupt after each instruction executes.
- When a MP is interrupted using Type 1 interrupt, it will execute one instruction and stop so that we can then examine the contents of registers and memory locations.
- In other words, in single step mode, a system will stop after it executes each instruction and waits for further direction from us.

3. **NON MASKABLE INTERRUPT : TYPE 2**
- A result of placing logic 1 on the NMI input pin causes type 2 interrupt.
- This input is non-maskable, which means that it cannot be disabled.
- Another common use of type 2 interrupt is to save program data in case of a system power failure or to deal with some other catastrophic failure conditions.
- Some external circuitry detects when the AC power to the system fails and sends an Interrupt signal to the NMI.

4. **BREAKPOINT INTERRUPT : TYPE 3**
- Used to insert a breakpoint into the program. When a break point is inserted, the system executes the instructions up to the breakpoint, and then stops execution.
- The INT 3 instruction is often used to store a breakpoint in a program for debugging.

5. **OVERFLOW INTERRUPT : TYPE 4**
- A special vector used with the INTO instruction.
- The 8086 overflow flag, OF, will be set if the signed result of an arithmetic operation on two signed numbers is too large to be represented in the destination register or memory location
- The INTO instruction interrupts the program if an overflow condition exists as reflected by overflow flag (OF).

PRIORITY OF 8086 INTERRUPTS

INTERRUPTS	**PRIORITY**
DIVIDE ERROR, INT n , INTO	HIGHEST
NMI	
INTR	
SINGLE STEP	LOWEST

INTERRUPT PRIORITY

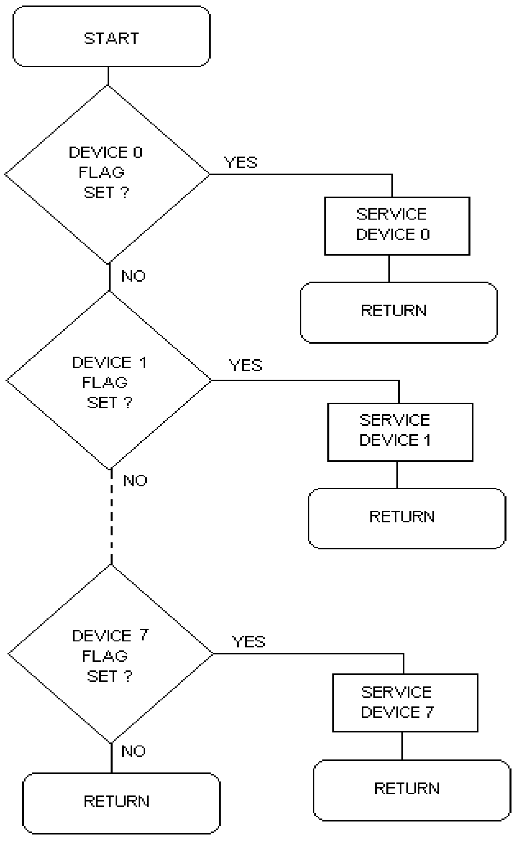
Multiple Interrupts

If more than one device is connected to the interrupt line, the processor needs to know to which device service routine it should branch to. The identification of the device requesting service can be done in either hardware or software, or a combination of both. The three main methods are:

1. Software Polling,
2. Hardware Polling, (Daisy Chain),
3. Hardware Identification (Vectored Interrupts).

1. Software Polling Determination of the Requesting Device

A software routine is used to identify the device requesting service. A simple polling technique is used, each device is checked to see if it was the one needing service. Having identified the device, the processor then branches to the appropriate interrupt-handling-routine address for the given device. The order in which the devices appear in the polling sequence determines their priority.



SOFTWARE POLLING FLOWCHART

Points to remember

1) it is wasteful of the processors time, as it needlessly checks the status of all devices all the time,

2) it is inherently slow, as it checks the status of all I/O devices before it comes back to check any given one again,

3) when fast devices are connected to a system, polling may simply not be fast enough to satisfy the minimum service requirements,

4) priority of the device is determined by the order in the polling loop, but it is possible to change it via software.

2. **Hardware Polling(Daisy Chain)**

This is significantly faster than a pure software approach. A daisy chain is used to identify the device requesting service.

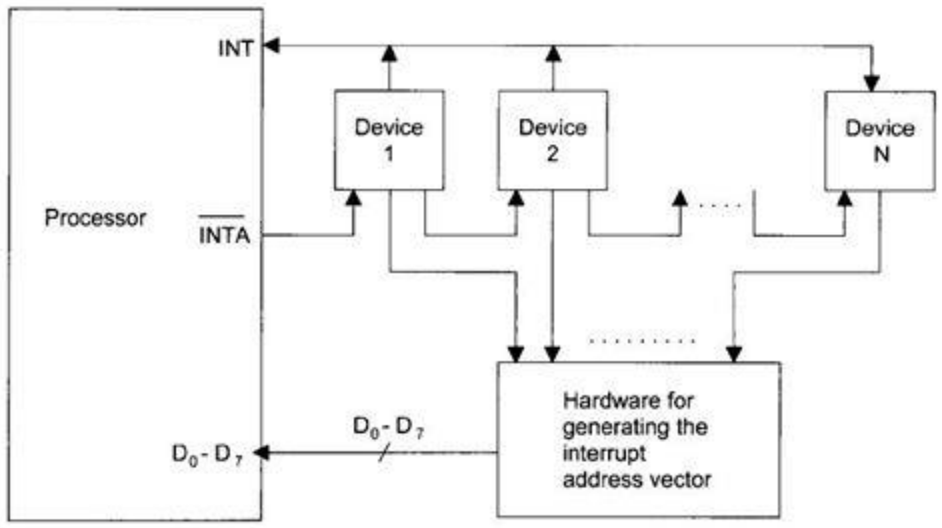


Fig: Daisy Chain Interrupt

Daisy chaining is used for level sensitive interrupts, which act like a wired 'OR' gate. Any requesting device can take the interrupt line low, and keep it asserted low until it is serviced.

Because more than one device can assert the shared interrupt line simultaneously, some method must be employed to ensure device priority. This is done using the interrupt acknowledge signal generated by the processor in response to an interrupt request.

Each device is connected to the same interrupt request line, but the interrupt acknowledge line is passed through each device, from the highest priority device first, to the lowest priority device last.

After preserving the required registers, the microprocessor generates an interrupt acknowledge signal. This is gated through each device. If device 1 generated the interrupt, it will place its identification signal on the data bus, which is read by the processor, and used to generate the address of the interrupt-service routine. If device 1 did not request the servicing, it will pass the interrupt acknowledge signal on to the next device in the chain. Device 2 follows the same procedure, and so on.

3. **Hardware Identification (Vectored Interrupts)**

This is the fastest system. The focus is placed on the requesting device to request the interrupt, and identify itself. The identity could be a branching address for the desired interrupt-handling routine.

If the device just supplies an identification number, this can be used in conjunction with a lookup table to determine the address of the required service routine. Response time is best when the device requesting service also supplies a branching address.

Priority Interrupt Controller Chips (PIC's) are hardware chips designed to make the task of a device presenting its own address to the CPU simple. The PIC also assesses the priority of the devices connected to it. Modern PIC's can also be programmed to prevent the generation of interrupts which are lower than a desired level.

The PIC arranges all devices in a list, devices given a lower priority are serviced when no other higher priority devices need servicing. This simplifies the software required to determine the device, resulting in an increase in speed.

The disadvantages are:

1) the extra chip required,

2) resultant increases in cost,

3) more board space and power consumption,

4) fixed priority in hardware.

Interrupt Processing

- The processor checks for interrupts. If interrupt has occurred, processor will complete the instruction currently being executed.
- The processor will disable the further interrupts.
- The processor stores the current state of program by PUSH operation i.e. the value of flag register and CS: IP will be stored into stack by PUSH operation.
- The processor will load the address of the ISR and execute the ISR. At the end of ISR, instruction IRET is used which makes the processor return from the ISR to the original program.

- After the execution of ISR, the processor restores the previous state of program i.e. it will restore the value of flag register and CS: IP from stack by POP operation.
- The processor enables the interrupts and then starts program execution from where it has been interrupted.

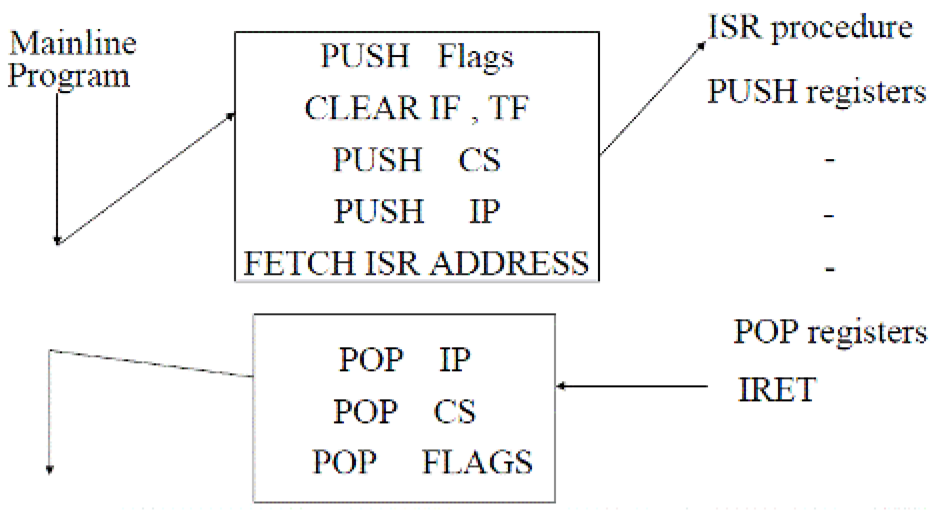


Fig: Interrupt Processing