

Chapter – 5

Interrupt Operations

- Interrupt is signals send by an external device to the processor, to request the processor to perform a particular task or work.
- Mainly in the microprocessor based system the interrupts are used for data transfer between the peripheral and the microprocessor.

Need for Interrupt: Interrupts are particularly useful when interfacing I/O devices that provide or require data at relatively low data transfer rate.

Interrupt Operations

The transfer of data between the microprocessor and input /output devices takes place using various modes of operations like programmed I/O, interrupt I/O and direct memory access. In programmed I/O, the processor has to wait for a long time until I/O module is ready for operation. So the performance of entire system degraded. To remove this problem CPU can issue an I/O command to the I/O module and then go to do some useful works. The I/O device will then interrupt the CPU to request service when it is ready to exchange data with CPU. In response to an interrupt, the microprocessor stops executing its current program and calls a procedure which services the interrupt.

The interrupt is a process of data transfer whereby an external device or a peripheral can inform the processor that it is ready for communication and it requests attention. The response to an interrupt request is directed or controlled by the microprocessor.

5.1 Polling versus Interrupt

- Polling the device usually means reading its status register every so often until the device's status changes to indicate that it has completed the request.
- Polling means the CPU keeps checking a flag to indicate if something happens.
- An interrupt driven device driver is one where the hardware device being controlled will cause a hardware interrupt to occur whenever it needs to be serviced.
- With interrupt, CPU is free to do other things, and when something happens, an interrupt is generated to notify the CPU. So it means the CPU does not need to check the flag.
- Polling is like picking up your phone every few seconds to see if you have a call. Interrupts are like waiting for the phone to ring.
- Interrupts win if processor has other work to do and event response time is not critical.
- Polling can be better if processor has to respond to an event ASAP; may be used in device controller that contains dedicated secondary processor.

Mechanisms to handle the interrupts for several devices

1. Polled interrupts

Polled interrupts are handled by using software which is slower than hardware interrupts. Here the processor has the general (common) interrupt service routine (ISR) for all devices. The priority of the devices is determined by the order in which the routine polls each device. The processor checks the starting with the highest priority device. Once it determines the source of the interrupt, it branches to the service routine for that device.

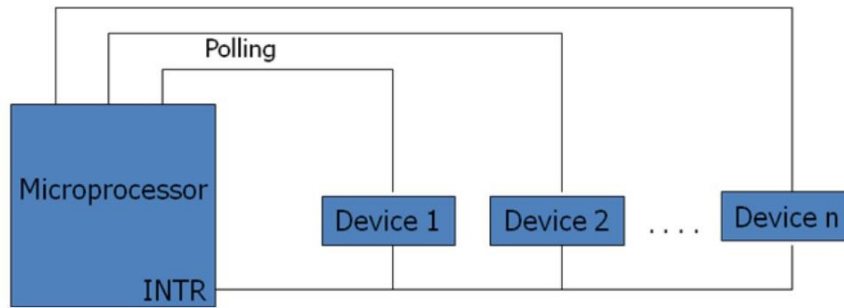


Fig: Polled Interrupt

Here several external devices are connected to a single interrupt line (INTR) of the microprocessor. When INTR signal goes up, the processor saves the contents of PC and other registers and then branches to an address defined by the manufactures of the processor.

2. Daisy chain (vectored) interrupt

In polled interrupt, the time required to poll each device may exceed the time to service the device through software. To improve this, the faster mechanism called vectored or daisy chain interrupt is used. Here the devices are connected in chain fashion. When INTR pin goes up, the processor saves its current status and then generates INTA signal to the highest priority device. If this device has generated the interrupt, it will accept the INTA; otherwise it will push INTA to the next priority device until INTA is accepted by the interrupting device. When INTA is accepted, the device provides a means to the processor for findings the interrupt address vector using external hardware.

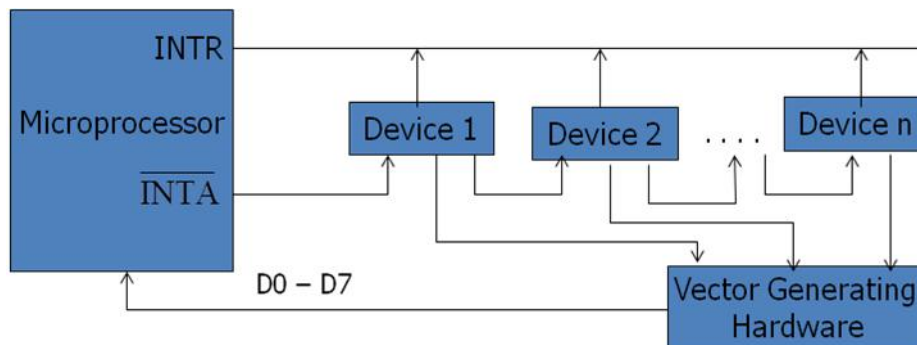


Fig: Vectored (Daisy Chain) Interrupt

5.2 Interrupt Processing Sequence

The occurrence of interrupt triggers a number of events, both in processor hardware and in software. The interrupt driven I/O operation takes the following steps.

- The I/O unit issues an interrupt signal to the processor for exchange of data between them.
- The processor finishes execution of the current instruction before responding to the interrupt.
- The processor sends an acknowledgement signal to the device that it issued the interrupt.
- The processor transfers its control to the requested routine called “Interrupt Service Routine (ISR)” by saving the contents of program status word (PSW) and program counter (PC).
- The processor now loads the PC with the location of interrupt service routine and the fetches the instructions. The result is transferred to the interrupt handler program.
- When interrupt processing is completed, the saved register’s value are retrieved from the stack and restored to the register.
- Finally it restores the PSW and PC values from the stack.

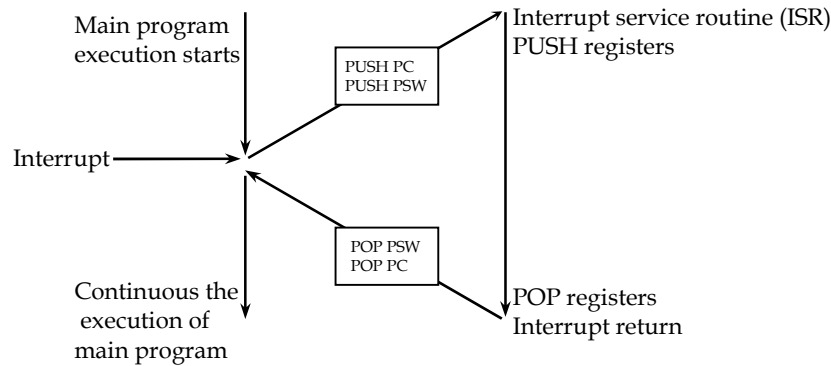


Fig: Interrupt Response for 8085 Microprocessor

Interrupt priority:

Microcomputers can transfer data to or from an external devices using interrupt through INTR pin. When device wants to communicate with the microcomputer, it connects to INTR pin and makes it high or low depending on microcomputer. The microcomputer responds by sending signal via its pin called interrupt acknowledgement \overline{INTA} . In differentiation with the occurrence of interrupts, basically following interrupts exist.

1. External interrupts:

These interrupts are initiated by external devices such as A/D converters and classified on following types.

- **Maskable interrupt :**
It can be enabled or disabled by executing instructions such as EI and DI. In 8085, EI sets the interrupt enable flip flop and enables the interrupt process. DI resets the interrupt enable flip flop and disables the interrupt.
- **Non-maskable interrupt:**
It has higher priority over maskable interrupt and cannot be enabled or disabled by the instructions.

2. Internal interrupts:

- These are indicated internally by exceptional conditions such as overflow, divide by zero, and execution of illegal op-code. The user usually writes a service routine to take correction measures and to provide an indication in order to inform the user that exceptional condition has occurred.
- These interrupts are used by using software to call the function of an operating system. Software interrupts are shorter than subroutine calls and they do not need the calling program to know the operating system's address in memory.

If the processor gets multiple interrupts, then we need to deal these interrupts one at a time and the dealing approaches are:

a. Sequential processing of interrupts

When user program is executing and an interrupt occurs interrupts are disabled immediately. After the interrupt service routine completes, interrupts are enabled before resuming the user program and the processor checks to see if additional interrupts have occurred.

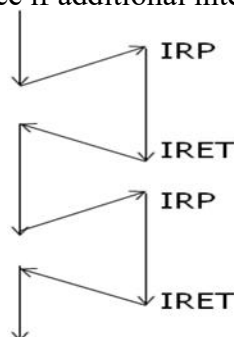


Fig: Sequential Interrupt Service

b. Priority wise processing of interrupts:

The drawback of sequential processing is that it does not take account of relative priority or time critical needs. The alternative form of this is to define priorities for interrupts and to allow an interrupt of higher priority to cause a lower priority interrupts pause until high priority interrupt completes its function.

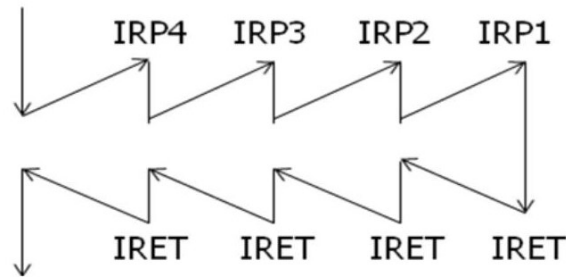


Fig: Priority wise Interrupt service

5.3 Interrupt Service Routine

- An interrupt service routine (ISR) is a software routine that hardware invokes in response to an interrupt.
- ISRs examine an interrupt and determine how to handle it.
- ISRs handle the interrupt, and then return a logical interrupt value.
- Its central purpose is to process the interrupt and then return control to the main program.

5.4 Interrupt Processing in 8085

8085 supports two types of interrupts.

1. Hardware
2. Software

Software interrupts:

The software interrupts are program instructions. These instructions are inserted at desired locations in a program.

The 8085 has eight software interrupts from RST 0 to RST 7. The vector address for these interrupts can be calculated as follows.

Interrupt number * 8 = vector address For RST 5; $5 * 8 = 40 = 28H$

Vector address for interrupt RST 5 is 0028H

The Table shows the vector addresses of all interrupts.

Interrupt	Vector address
RST 0	0000 _H
RST 1	0008 _H
RST 2	0010 _H
RST 3	0018 _H
RST 4	0020 _H
RST 5	0028 _H
RST 6	0030 _H
RST 7	0038 _H

5.4.1 Interrupt Pins and Priorities (Hardware interrupts)

An external device initiates the hardware interrupts and placing an appropriate signal at the interrupt pin of the processor.

If the interrupt is accepted then the processor executes an interrupt service routine. The 8085 has five hardware interrupts

(1) TRAP (2) RST 7.5 (3) RST 6.5 (4) RST 5.5 (5) INTR

Interrupt type	Trigger	Priority	Maskable	Vector address
TRAP	Edge and Level	1 st	No	0024H
RST 7.5	Edge	2 nd	Yes	003CH
RST 6.5	Level	3 rd	Yes	0034H
RST 5.5	Level	4 th	Yes	002CH
INTR	Level	5 th	Yes	-

TRAP:

- This interrupt is a non-maskable interrupt.
- Used for operating microprocessor in single step mode and hence important in debugging.
- TRAP has the highest priority and is vectored interrupt.
- TRAP interrupt is edge and level triggered. This means that TRAP must go high and remain high until it is acknowledged.

RST 7.5:

- RST 7.5 interrupt is a maskable interrupt.
- It has the second highest priority.
- It is edge sensitive. ie. Input goes to high and no need to maintain high state until it recognized.
- It is disabled by,
 1. DI instruction
 2. System or processor reset.
 3. After reorganization of interrupt.
- Enabled by EI instruction.

RST 6.5 and 5.5:

- The RST 6.5 and RST 5.5 both are level triggered. . ie. Input goes to high and stays high until it recognized.
- Maskable interrupt. It is disabled by,
 1. DI, SIM instruction
 2. System or processor reset.
 3. After reorganization of interrupt.
- Enabled by EI instruction.
- The RST 6.5 has the third priority whereas RST 5.5 has the fourth priority.

INTR:

- INTR is a maskable interrupt.
- It is disabled by,
 1. DI, SIM instruction
 2. System or processor reset.
 3. After reorganization of interrupt.
- Enabled by EI instruction.

- Non- vectored interrupt. After receiving INTA (active low) signal, it has to supply the address of ISR.
- It has lowest priority.
- It is a level sensitive interrupts. ie. Input goes to high and it is necessary to maintain high state until it recognized.
- The following sequence of events occurs when INTR signal goes high.
 1. The 8085 checks the status of INTR signal during execution of each instruction.
 2. If INTR signal is high, then 8085 complete its current instruction and sends active low interrupt acknowledge signal, if the interrupt is enabled.
 3. In response to acknowledge signal, external logic places an instruction OPCODE on the data bus.
 4. On receiving the instruction, the 8085 save the address of next instruction on stack and execute received instruction.

5.4.2 Using Programmable Interrupt Controller (PIC)

Priority interrupt controller (PIC)

The INTR pin can be used for multiple peripherals and to determine priorities among these devices when two or more peripherals request interrupt service simultaneously, PIC is used. If there are simultaneous requests, the priorities are determined by the encoder, it responds to the higher level input, ignoring the lower level input. The drawback of the scheme is that the interrupting device connected to input I_7 always has the highest priority. The PIC includes a status register and a priority comparator in addition to a priority encoder.

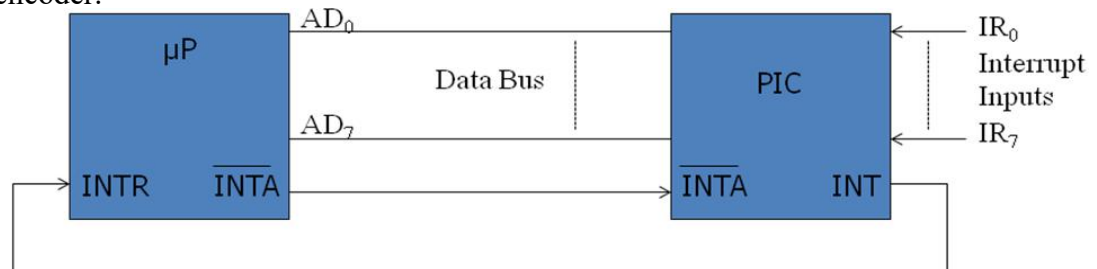
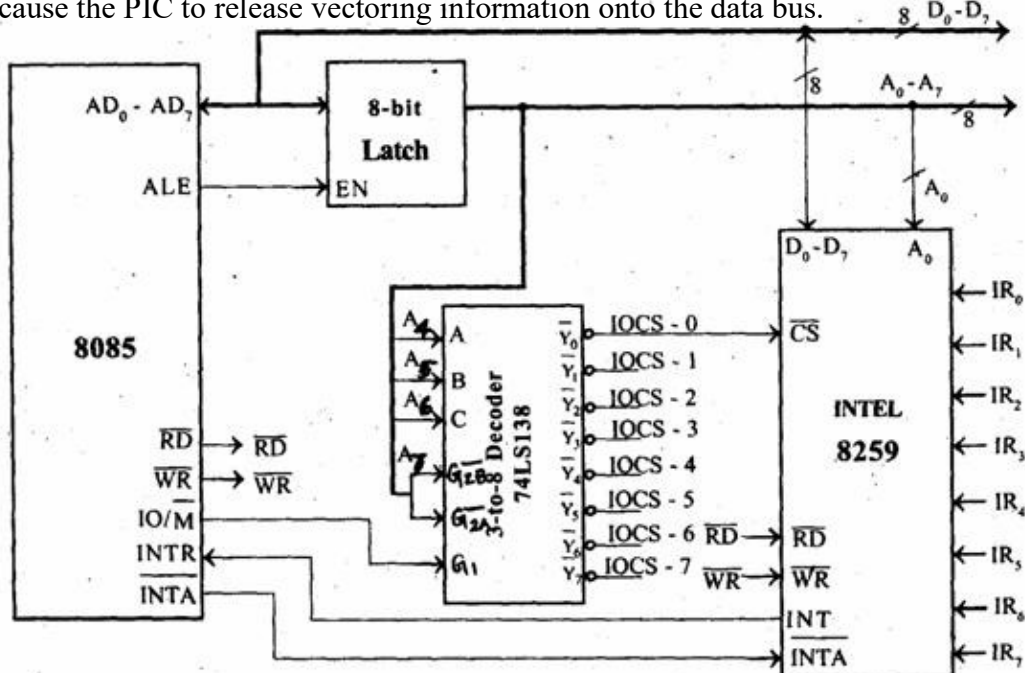


Fig: Multiple Interrupts using PIC

Today this device is replaced by a more versatile one called a programmable interrupt controller 8259A. When an 8259A receives an interrupt signal on one of its IR inputs, it sends an interrupt request signal to the INTR input of the μP . Then INTA pulses will cause the PIC to release vectoring information onto the data bus.



- It requires two internal address and they are $A=0$ or $A=1$.
- It can be either memory mapped or I/O mapped in the system. The interfacing of 8259 to 8085 is shown in figure is I/O mapped in the system.
- The low order data bus lines D0-D7 are connected to D0-D7 of 8259.
- The address line A0 of the 8085 processor is connected to A0 of 8259 to provide the internal address.
- The 8259 require one chip select signal. Using 3-to-8 decoder generates the chip select signal for 8259.
- The address lines A4, A5 and A6 are used as input to decoder.
- The control signal IO/M (low) is used as logic high enables for decoder and the address line A7 is used as logic low enable for decoder.
- The I/O addresses of 8259 are shown in table below.

	Binary Address								Hexa address
	Decoder input/ enable			Input to address pin of 8259					
	A ₇	A ₆	A ₅	A ₄	A ₃	A ₂	A ₁	A ₀	
For A ₀ of 8259 to be zero	0	0	0	0	x	x	x	0	00
For A ₀ of 8259 to be one	0	0	0	0	x	x	x	1	01

Note : Don't care "x" is considered as zero.

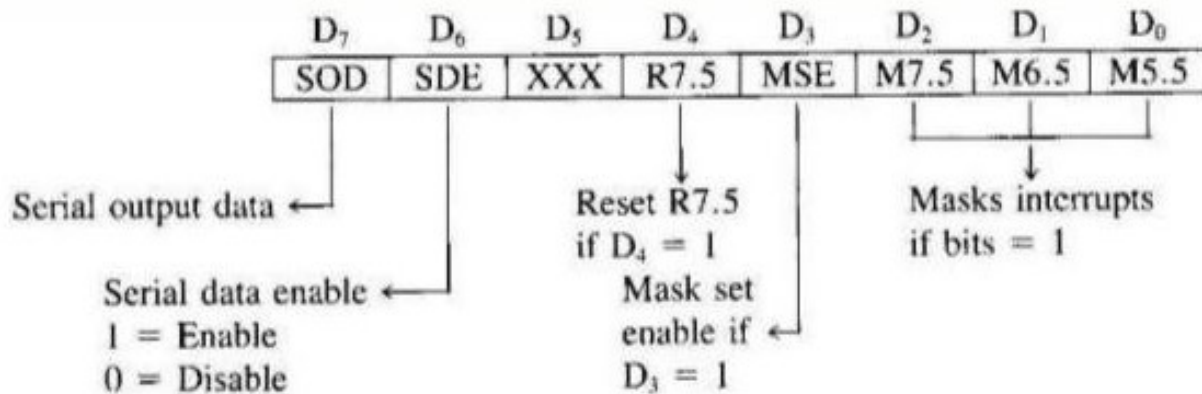
Working of 8259 with 8085 processor:

- First the 8259 should be programmed by sending Initialization Command Word (ICW) and Operational Command Word (OCW). These command words will inform 8259 about the following,
 1. Type of interrupt signal (Level triggered / Edge triggered).
 2. Type of processor (8085/8086).
 3. Call address and its interval (4 or 8)
 4. Masking of interrupts.
 5. Priority of interrupts.
 6. Type of end of interrupts.
- Once 8259 is programmed it is ready for accepting interrupt signal. When it receives an interrupt through any one of the interrupt lines IR0-IR7 it checks for its priority and also checks whether it is masked or not.
- If the previous interrupt is completed and if the current request has highest priority and unmasked, then it is serviced.
- For servicing this interrupt the 8259 will send INT signal to INTR pin of 8085.
- In response it expects an acknowledge INTA (low) from the processor.
- When the processor accepts the interrupt, it sends three INTA (low) one by one.
- In response to first, second and third INTA (low) signals, the 8259 will supply CALL op-code, low byte of call address and high byte of call address respectively. Once the processor receives the call op-code and its address, it saves the content of program counter (PC) in stack and loads the CALL address in PC and start executing the interrupt service routine stored in this call address.

5.4.3 Interrupt instructions

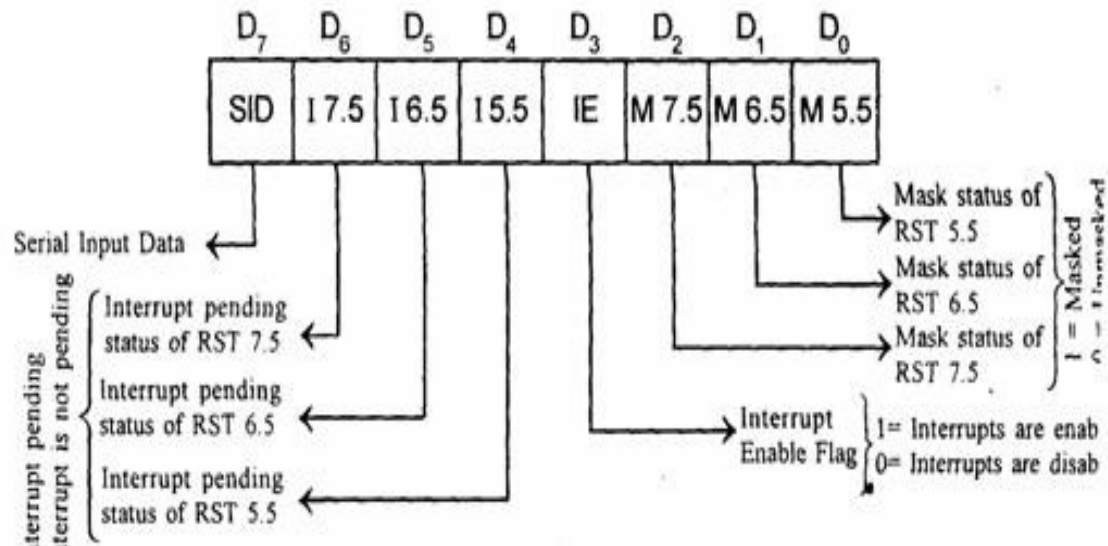
SIM instruction:

- The 8085 provide additional masking facility for RST 7.5, RST 6.5 and RST 5.5 using SIM instruction.
- This is a multipurpose instruction and used to implement the 8085 interrupts 7.5, 6.5, 5.5, and serial data output.
- The masking or unmasking of RST 7.5, RST 6.5 and RST 5.5 interrupts can be performed by moving an 8-bit data to accumulator and then executing SIM instruction.
- The format of the 8-bit data is shown below.



- ☐ SOD—Serial Output Data: Bit D₇ of the accumulator is latched into the SOD output line and made available to a serial peripheral if bit D₆ = 1.
- ☐ SDE—Serial Data Enable: If this bit = 1, it enables the serial output. To implement serial output, this bit needs to be enabled.
- ☐ XXX—Don't Care
- ☐ R7.5—Reset RST 7.5: If this bit = 1, RST 7.5 flip-flop is reset. This is an additional control to reset RST 7.5.
- ☐ MSE—Mask Set Enable: If this bit is high, it enables the functions of bits D₂, D₁, D₀. This is a master control over all the interrupt masking bits. If this bit is low, bits D₂, D₁, and D₀ do not have any effect on the masks.
- ☐ M7.5—D₂ = 0, RST 7.5 is enabled.
 = 1, RST 7.5 is masked or disabled.
- ☐ M6.5—D₁ = 0, RST 6.5 is enabled.
 = 1, RST 6.5 is masked or disabled.
- ☐ M5.5—D₀ = 0, RST 5.5 is enabled.
 = 1, RST 5.5 is masked or disabled.

RIM instruction



- The status of pending interrupts can be read from accumulator after executing RIM instruction.
- This is a multipurpose instruction used to read the status of RST 7.5, 6.5, 5.5 and read serial data input bit.
- When RIM instruction is executed an 8-bit data is loaded in accumulator, which can be interpreted as shown in above fig.
- Bits 0-2 show the current setting of the mask for each of RST 7.5, RST 6.5 and RST 5.5. They return the contents of the three masks flip flops. They can be used by a program to read the mask settings in order to modify only the right mask.
- Bit 3 shows whether the maskable interrupt process is enabled or not. It returns the contents of the Interrupt Enable Flip Flop. It can be used by a program to determine whether or not interrupts are enabled.
- Bits 4-6 show whether or not there are pending interrupts on RST 7.5, RST 6.5, and RST 5.5. Bits 4 and 5 return the current value of the RST5.5 and RST6.5 pins. Bit 6 returns the current value of the RST7.5 memory flip flop.
- Bit 7 is used for Serial Data Input. The RIM instruction reads the value of the SID pin on the microprocessor and returns it in this bit.

DI

- Disable interrupts
- The interrupt enable flip-flop is reset and all the interrupts except the TRAP are disabled. No flags are affected.
- 1 byte instruction

EI

- Enable interrupts
- The interrupt enable flip-flop is set and all interrupts are enabled.
- No flags are affected.
- After a system reset or the acknowledgement of an interrupt, the interrupt enable flip flop is reset, thus disabling the interrupts.
- This instruction is necessary to enable the interrupts (except TRAP).
- 1 byte instruction

5.5 Interrupt Processing in 8086

The meaning of 'interrupts' is to break the sequence of operation. While the CPU is executing a program, on 'interrupt' breaks the normal sequence of execution of instructions, diverts its execution to some other program called Interrupt Service Routine (ISR). After executing ISR, the control is transferred back again to the main program. Interrupt processing is an alternative to polling.

Interrupt Response sequence

- It decrements SP by 2 and pushes the flag register on the stack.
- Disables INTR by clearing the IF.
- It resets the TF in the flag Register.
- It decrements SP by 2 and pushes CS on the stack.
- It decrements SP by 2 and pushes IP on the stack.
- Fetch the ISR address from the interrupt vector table.

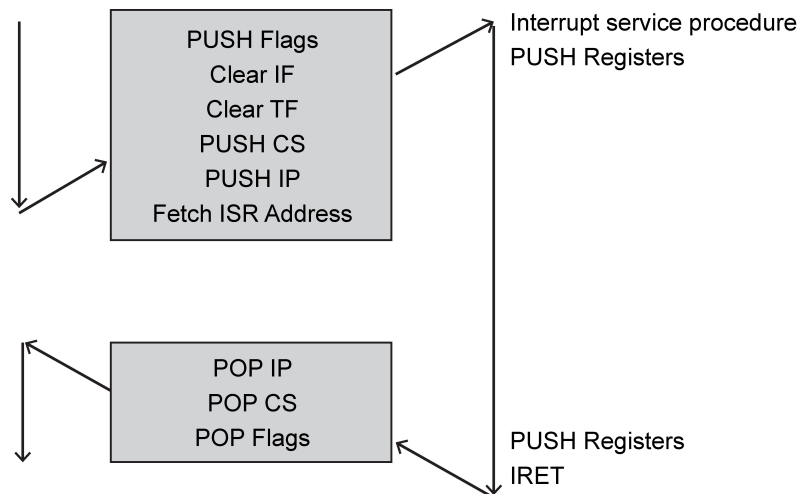


Fig: Interrupt Response for 8086 microprocessor

5.5.1 Interrupt Pins

INTR and NMI

- **INTR** is a maskable hardware interrupt. The interrupt can be enabled/disabled using STI/CLI instructions or using more complicated method of updating the FLAGS register with the help of the POPF instruction.
- **NMI** is a non-maskable interrupt. Interrupt is processed in the same way as the INTR interrupt. Interrupt type of the NMI is 2, i.e. the address of the NMI processing routine is stored in location 0008h. This interrupt has higher priority than the maskable interrupt.

5.5.2 Interrupt Vector Table and its Organization

- An interrupt vector is a pointer to where the ISR is stored in memory.
- All interrupts (vectored or otherwise) are mapped onto a memory area called the Interrupt Vector Table (IVT).
- Interrupt Vector Table (IVT) is a 1024 bytes sized table that contains addresses of interrupts. Each address is of 4 bytes long of the form offset:segment, which represents the address of a routine to be called when the CPU receives an interrupt.
- The purpose of the IVT is to hold the vectors that redirect the microprocessor to the right place when an interrupt arrives.
- The lowest five types are dedicated to specific interrupts.
- Interrupt 5 to 31 are reserved by Intel for use in more complex microprocessor.

- The upper 224 interrupt types from 32 to 255 are available to use for hardware and

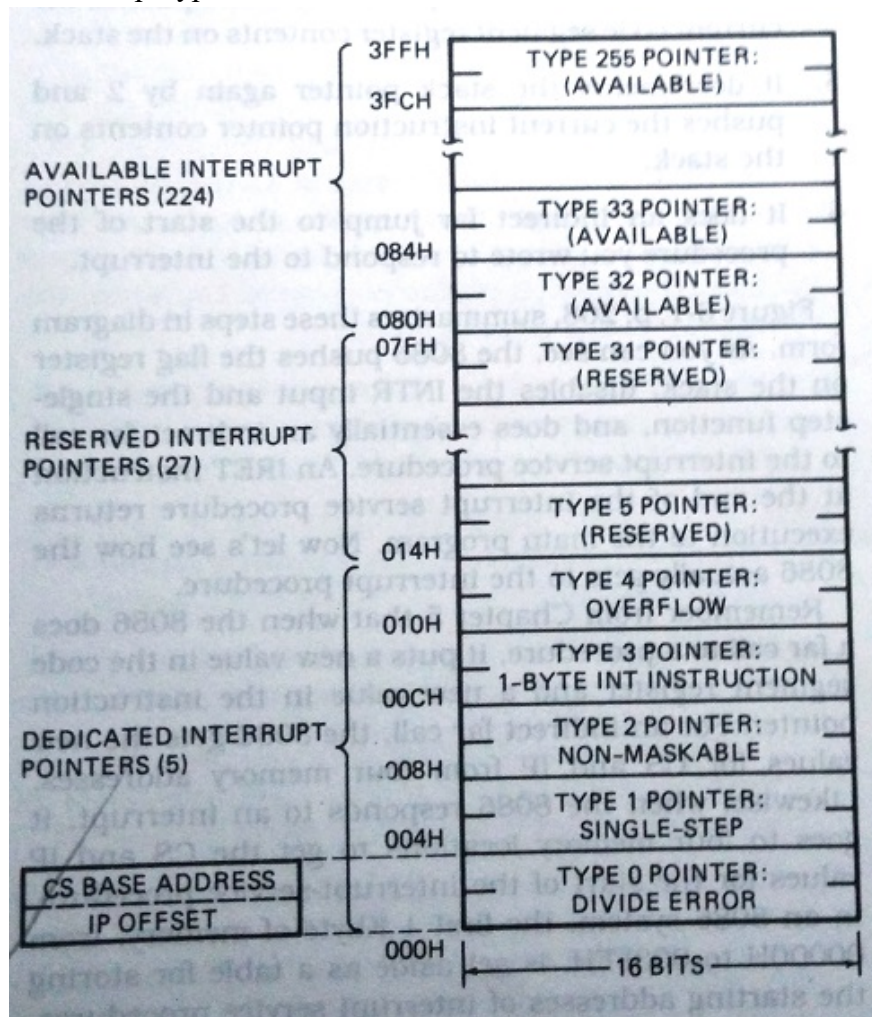


Fig:

IVT Structure (Organization)

8086 Interrupt Types

Type 0 (divide error Interrupt)

- INT00 is invoked by the microprocessor whenever there is an attempt to divide a number by zero.
- Cannot be disabled in any way

Type 1 (single step Interrupt)

- For single stepping, the trap flag must be 1
- After execution of each instruction, 8086 automatically jumps to 00004H to fetch 4 bytes for CS: IP of the ISR.
- Used for debugging of program

Type 2 (Non maskable Interrupt)

- Whenever NMI pin of the 8086 is activated by a high signal (5v), the CPU jumps to physical memory location 00008 to fetch CS:IP of the ISR associated with NMI.

Type 3 (breakpoint Interrupt)

- Type 3 interrupt is produced by the execution of the INT instruction. The main use of type 3 interrupt is to implement a breakpoint function in the system

Type 4 (Signed number overflow Interrupt)

- There is an instruction associated with this INTO (interrupt on overflow).
- If INTO is placed after a signed number arithmetic as IMUL or ADD the CPU will activate INT 04 if OF = 1.
- In case where OF = 0, the INTO is not executed but is bypassed and acts as a NOP.

5.5.3 Software and Hardware Interrupt

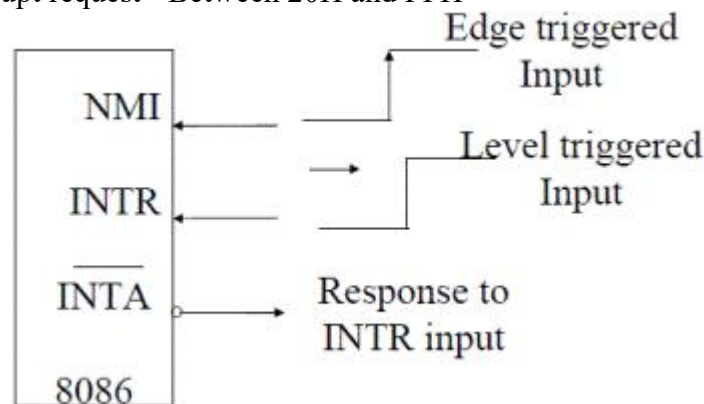
Types of Interrupts: There are two types of Interrupts in 8086. They are:

(i) Hardware Interrupts (External Interrupts). The Intel microprocessors support hardware interrupts through:

- Two pins that allow interrupt requests, INTR and NMI
- One pin that acknowledges, INTA, the interrupt requested on INTR.

Performance of Hardware Interrupts

- NMI : Non maskable interrupts - TYPE 2 Interrupt
- INTR : Interrupt request - Between 20H and FFH



(ii) Software Interrupts (Internal Interrupts and Instructions) .Software interrupts can be caused by:

- INT instruction - breakpoint interrupt. This is a type 3 interrupt.
- INT <interrupt number> instruction - any one interrupt from available 256 interrupts.
- INTO instruction - interrupt on overflow
- Single-step interrupt - generated if the TF flag is set. This is a type 1 interrupt. When the CPU processes this interrupt it clears TF flag before calling the interrupt processing routine.
- Processor exceptions: Divide Error (Type 0), Unused Opcode (type 6) and Escape opcode (type 7).
- Software interrupt processing is the same as for the hardware interrupts.
- - Ex: INT n (Software Instructions)

5.5.4 Interrupt Priorities

Interrupt	Priority
Divide Error, INT(n),INTO	Highest
NMI	↓
INTR	
Single Step	
	Lowest