Procedure

eg: factorial Program:

```
MOV  cl, N
MOV  Al, 01h
call Proce
MOV  [1000], Ax.
Jmp  l
Proce: Mul  cl
       loop Proce
       RET

l: Int 03h.
```

disadvantage:

1. Takes more time to execute.
2. Used for large sub program.

Advantage

1. occupy less memory.

* Procedure uses "CALL" and "RET" Instructions.
* when "CALL label/address" Instruction executes, : the
  1. control transfers to the sub program.
  2. "IP", flag status, "CS" Register stored into stack for return to next instruction.
* In sub program when "RET" instruction executes the "IP, CS & flag status" retrieves from stack, and control transfers to the next instruction.

MACRO:-

```
Adder Macro
  Add AL, BL
  Mov [2000], AL
endm
UP: mov AL, N1
    mov BL, N2
    Adder
    Jmp UP
    INT 03
```

When ever we need to use a group of instructions several times through out a program, there are two ways, we can avoid having to write the group of inst. each time we want to use it. one way is to write the group of inst. as a seperate procedure. and another is MACRO.

When the repeated group of inst. is too short or not appropriate to be written as a proc. we use a Macro. A MACR is a group of inst. at the starting of our program. Every time you see a Macro name in the program, the assembler replaces it with the group of inst. defined as the macro. at the start of the prog.

Dis Advantage of using MACRO is the Assembler generates machine codes for the group of ins. each time the MACRO called, this will make the prog. take up more memory than procedure.

Using a MACRO avoids the overhead time involved in calling and returning from a procedure.

(25) Interrupts of 8086 :

(14)

* The dictionary meaning of the word 'interrupt' is to break the sequence of operation. While the CPU is executing a program, an '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 which was being executed at the time of interruption.

* Broadly, there are two types of interrupts:
  1, External interrupt and 2, internal interrupt.

External interrupt : An external device or a signal, interrupts the processor from outside. eg: a keyboard interrupt.

Internal Interrupt : Is generated internally by the Precessor circuit, or by the execution of an interrupt instruction.
eg: divide by zero interrupt, overflow interrupt etc.,

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

* The 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, however, may be masked using the Interrupt Flag (IF).

Interrupt structure of 8086:

An 8086 interrupt can come from any one of three sources.

One source is an exteral signal applied to the NMI input pin or to the INTR i/p pin. An interrupt caused by a signal applied to one of these inputs is refersed to as a hardware interrupt.

A second source of an interrupt is execution of the interrupt instruction, INT. this is refersed as s/w int.

The third source of an interrupt is some error condition produced in the 8086 by the execution of an instruction. An example of this is the divide by zero, the 8086 will automatically interrupts the current execution.

At the end of each instruction cycle, the 8086 checks to see if any interrupts have been requested. If an interrupt has been requested, the 8086 responds to the interrupt by stepping through the following series of steps:

1) It decrements the stack pointer by 2 and pushes the flag register on the stack.
2) It disables the 8086 INTR interrupt input by clearing the IF in the flag register
3) It reset the TF in the flag register.
4) It decrements the SP by 2 and pushes the current code segment register contents on the stack.

5, It decrements the SP again by 2 and pushes the current IP contents on the stack

6, It does an indirect far Jump to ISR.

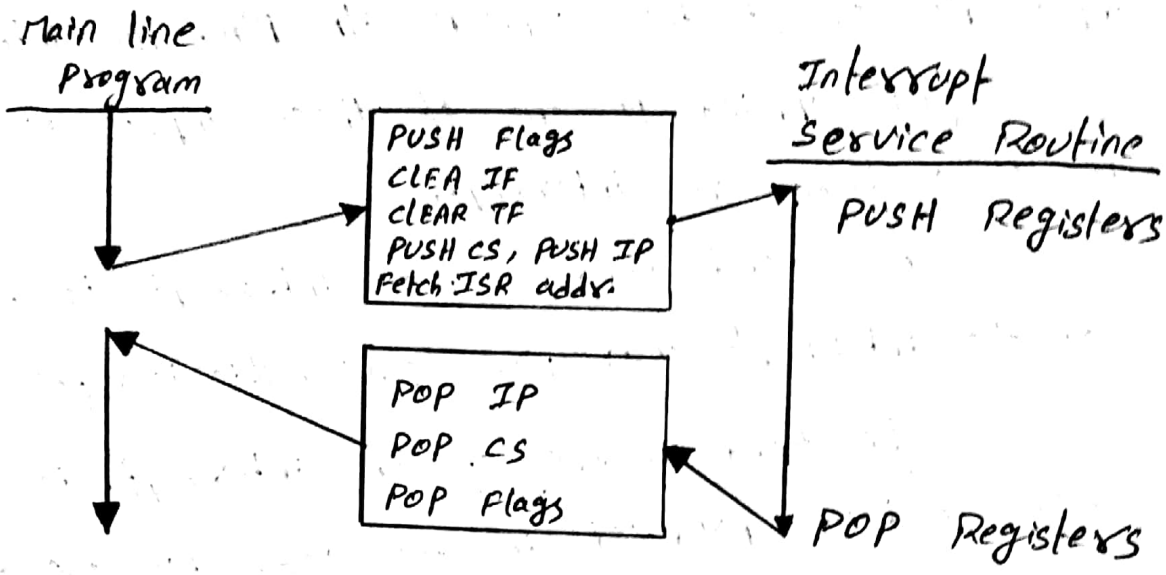

Fig: Interrupt Responce of 8086.

7, An IRET instruction at the end of the ISR returns execution to the main program.

Vector interrupt Table :

When the 8086 responds to an interrupt, it goes to four memory locations to get the CS and IP values for the start of the ISR.

In an 8086 system, the first 1Kbyte of memory from 00000H to 003FFH, is set aside as a table for storing the starting addresses of ISR. Since 4 bytes are required to store the CS and IP values for each ISR, the table can hold the starting addresses for up to 256 interrupt procedures. The starting addres of an ISR is often called the interrupt vector or the Interrupt pointer, so the table is refered to as the Interrupt-vector table (or) the interrupt pointer table.

The instruction pointer value is put in the low word of the vector, and the code segment register is put in the high word of the vector. Each double word interrupt vector is identified by a number 0 to 255. Intel calls this number the 'type' of the interrupt.

The lowest five types are dedicated to specific interrupts, such as the divide by zero interrupt, the single step interrupt, and the NMI. Interrupt types 5 to 31 are reserved by Intel for use in more complex microprocessors, such as the 80286, 80386 and 80486. The upper 224 interrupt types, from 32 to 255, are available to user for hardware or software interrupts.

As the vector for each interrupt type requires four memory locations. therefore, when the 8086 responds to a particular type interrupt, it automatically multiplies the type by 4 to produce the desired address in the vector table. It then goes to that address in the table to get the starting addres of ISR and then it executes the ISR..

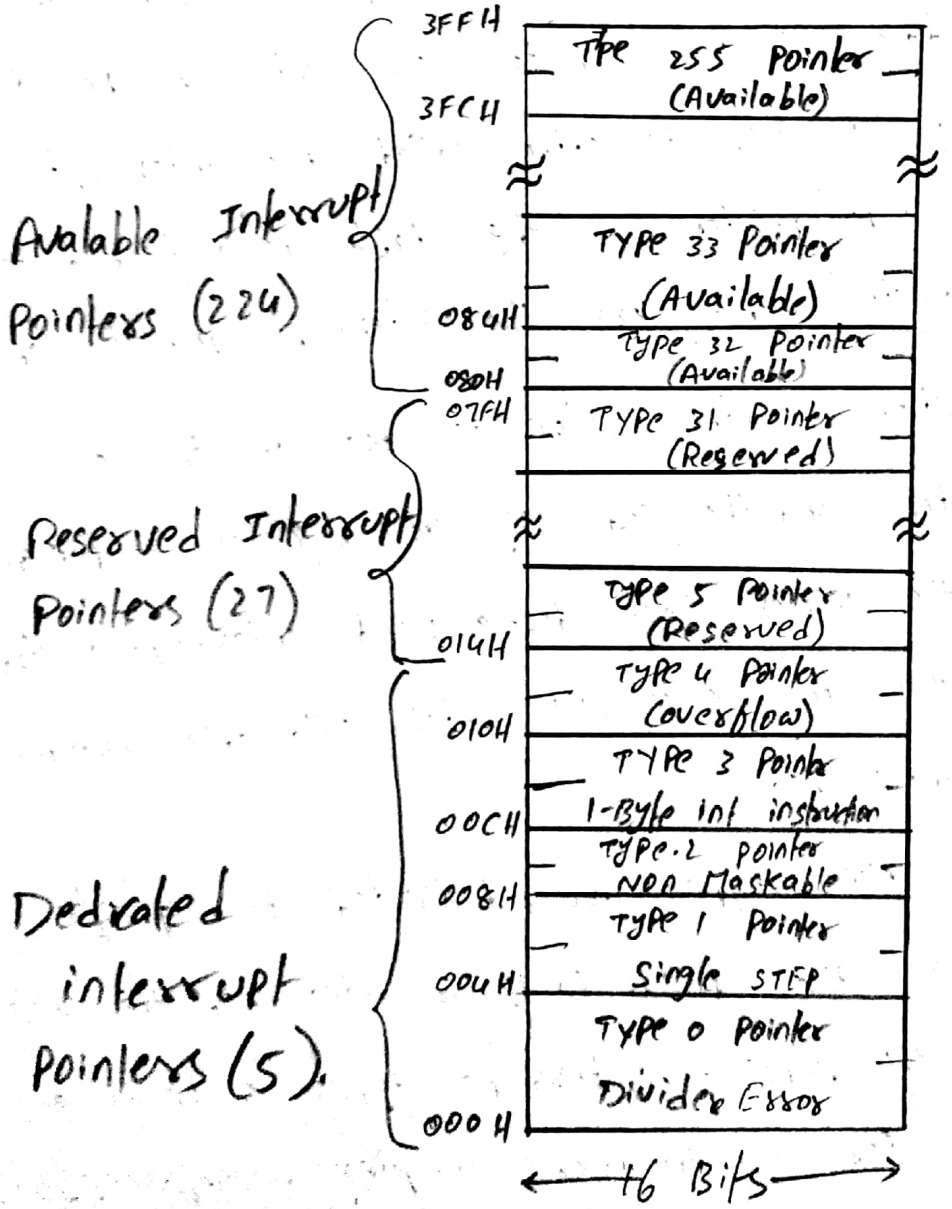

fig: 8086 interrupt - Pointer table.

8086 Interrupt Types : 8086 Interrupt Response (Service Routine)

Divide by zero Interrupt - Type 0

The 8086 will automatically do a type 0 interrupt if the result of a DIV operation or an IDIV operation is too large to fit in the destination register.

For the type 0 interrupt, the 8086 pushes the flag register on the stack, resets IF & TF and pushes return address CS & IP on the stack. It then gets the new IP & CS values for the start of interrupt service procedure from address 00000h & 00002h respectively. in the interrupt pointer table.

single step Interrupt - Type-1 :

In single step mode, the system will stop after it executes each instruction and wait for further direction. The 8086 Trap flag & Type 1 int. response make it quite easy to implement a single step feature in an 8086 based system. If the 8086 Trap flag is set, the 8086 will automatically do a type 1 interrupt after each instruction executes.

For the typ 1 interrupt, the 8086 pushes flag on stack, resets IF & TF, & pushes CS & IP on stack. It then gets the new IP & CS values for start of ISR from addresses 00004 & 00006 respectively in the interrupt pointer table.

Non Maskable Interrupt - Type-2

The name non maskable given to this input pin on the 8086 means that the type2 interrupt response cannot be disabled by any program instruction. so

This signal used for ex highest priority external hardware must be taken care of.

Another common use of the type2 interrupt is to save program data in case of a system power failure.

the 8086 will automatically do a type2 interrupt response when it receives a low to high transition on its NMI pin.

For the type2 interrupt, the 8086 pushes flag on stack, resets IF and TF, pushes CS & IP on stack. It then gets new IP and CS for start of ISR from addresses 00008h and 0000Ah from Interrupt pointer table.

overflow Interrupt - Type-4

The 8086 overflow flag will be set if the signed result of an arithmetic operation on two signed numbers is too large to be reperesented in the destination register.

The way of detecting and responding to an overflow error is to put INTO (~~0CE~~) ~~3E~~, immediately after the arithmetic instruction in the program. If overflow flag is not set then INTO simply function as an NOP. However if the overflow flag is set, indicating overflow error, the 8080 will do a INT 4 interrupt.

For the type 4 interrupt, the 8086 pusher flag on