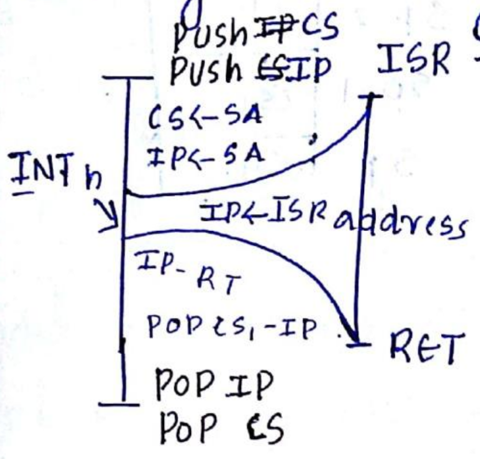
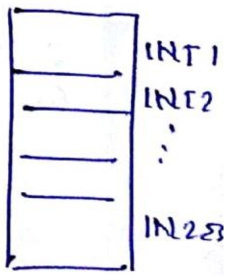
Interrupts

Interrupts is a special condition that occurs during working of up.



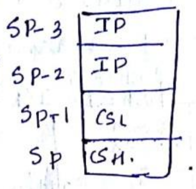
- 0 - 255 interrupts are available in 8086.
- Whenever a interrupt enters to a running progm. Then the INTR is given to ISR here INTR is solved and return to next instruction to be executed.
- 1 INTR needs 4 location ∴ 4×256 = 1kb.
- In 1MB m/y 1kb is ~~us~~ reserved for ISR
- IP- next address, RET contains address of the next executing instruction.



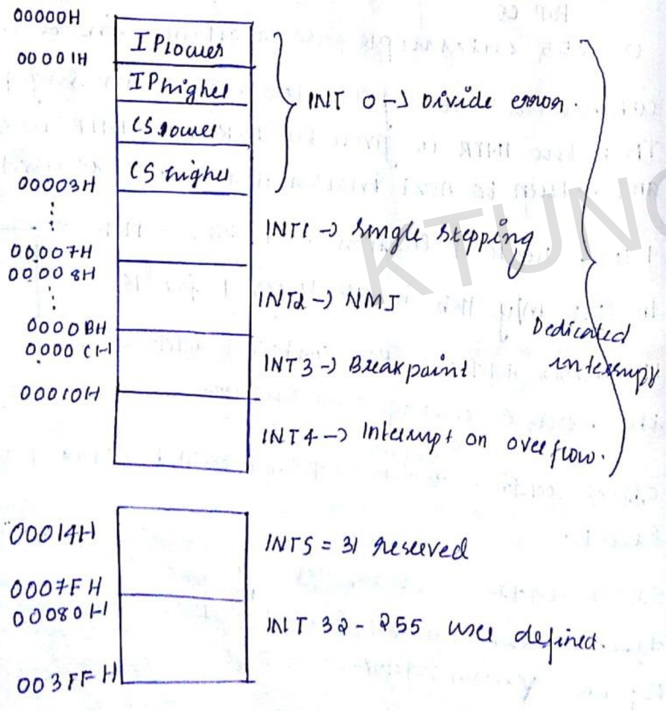
- Offset address, ~~return~~ Segment address will be stored in stack.
- Stack address= Return address
- ~~After~~ Return address loaded to stack → load IP address → ISR executes → Pop the return address.
- Here we need two memory.

segment address is also needed while two are programs.

- Segment address have more priority ∴ it is push to higher address of the stack.
- pop is ~~reversed to~~ of push.



Interrupt Vector Table:



~~the~~ ISR is not uniformly/sequentially stored ∴ to get the needed interrupt we need to know the location of the interrupt. location of all these instructions are stored in IVT using this address we identify the ISR.
Address of the ~~ad~~ interrupt location is always fixed.

8086 interrupt

An interrupt is a special condition that arises during the working of up The up services it by executing a subroutine called Interrupt Service Routine (ISR).

- There are three sources of interrupts for 8086.

External signal (H/w interrupts)

These interrupts occur as a signal on the external pins of the up. 8086 has ~~2~~ two pins to accept h/w interrupts NMI & INTR

Special instruction (S/w interrupts

These interrupts caused by writing the s/w interrupt instructⁿ INTn, where n can be any value from 0-255. Hence all 256 interrupts can be invoked By software. condition produced by the programs. (internally generated interrupts) 8086 is interrupted when some special condition occurs while executing

certain instructns in the prgm. Eg: an error
c. in division automatically causes INT O inter-
rupt.

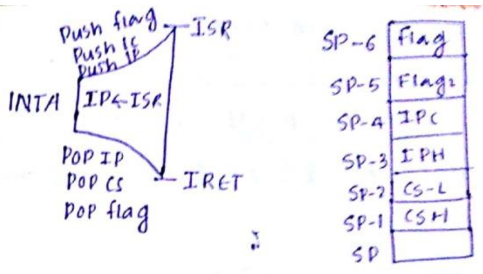
Interrupt Vector Table.

The IVT contains ISR address for the 256 address. Each ISR address is stored as CS and IP. As each ISR address is of 4 bytes (2CS and 2IP). Each ISR address requires 4 location to be stored. There are 256 ~~addresse~~ interrupts INT0 – INT255. ∴ the total size of IVT is 206×4 = 1KM. The 1st 1 KB of Mly 0000H .. 0003FFH are reserved for IVT. whereever any interrupt INTm occurs µp does N×4 to get values of IP and CS from the IVT and hence perform the ISR

maskable: It can hide the interrupt
non-maskable: It is a single non masnable interrupt having higher priority than the maskable interrupt request pin (INTR)

ACTIONS

1. complete the current instruction that is in progress
2. Push the flag reg value onto the stack
3. Pushes IC and IP value of the return address onto stack.
4. IP is loaded from the contents of word location 00008H.



5. CS is loaded from the content of the next word location 0000AH
6. Interrupt flag and trap flag are reset to 0.

APPLICATION

1. It is used to during power failure.
2. It is used to during critical response
3. non recoverable between errors

INTR

INTR is a maskable interrupt because the µp will be interrupted only if interrupts are enable using set interrupt flag instructions it should be unable using clear interrupt flag instruction

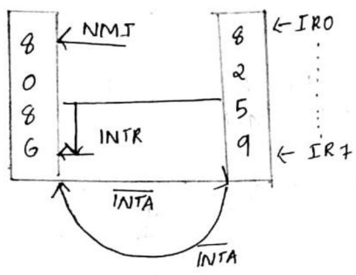
- The INTR interrupt is activated by I/O port.

% Programmable Interrupt Controller 8259 (PIC)

→ Features

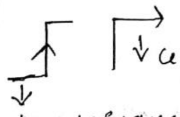
- It is used to increase the no: of interrupts
- can handle edge as well as level triggered.
- Flexible priority structure.
- can be invoked individualy

- Vector address is programmab
- 8259 compulsory initialized (vector no, mask, trigger, priority)
- Single configuration and cascaded configu



NMI, INTR these pins handle or control the interrupt

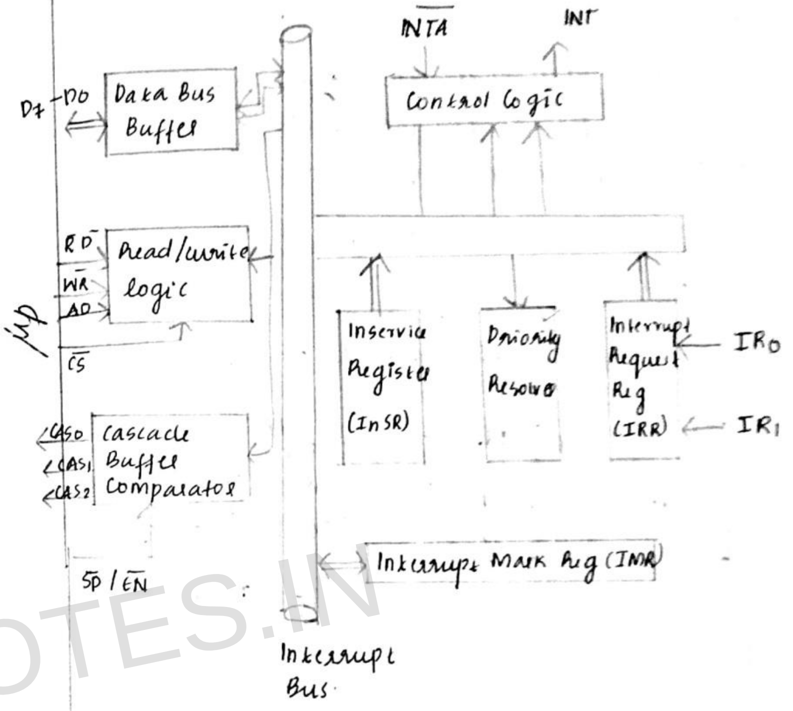
- Maskable: disable interrupt
- (vectored) NMI → INTERRUPT (it goes to location)
- Multiple interrupt will be control the 8259 chip
- 8259 → have 8 interrupt.
- It will send vector number



level triggered. IR_0 → highest priority.

edge triggered

- Single configuration: only one 8259 IC chip will be used
- cascaded configuration: one or more 8259 IC chip will be used.



8259 Architecture

IMR: It can helps to disable the interrupt

up: send the data's through the data bus

SP / EN: It helps to active slave

SP = 0 slave active

8259 interrupt the uP

INTA is used to able the vectored table

up send the acknowledgment to 8259

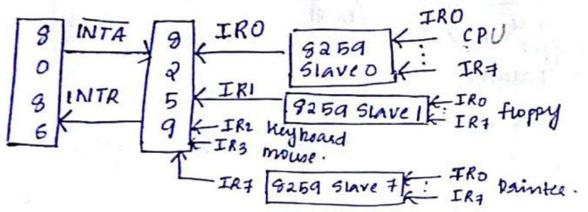
calculate the vector no and the up [illegible] the no: to 8259

then it goes to IVT then get the memory then it goes to the ISR it gives memory and then it will execute.

IRR → Store the interrupts

Cascaded Interrupt:

Expand the interrupt of 8259



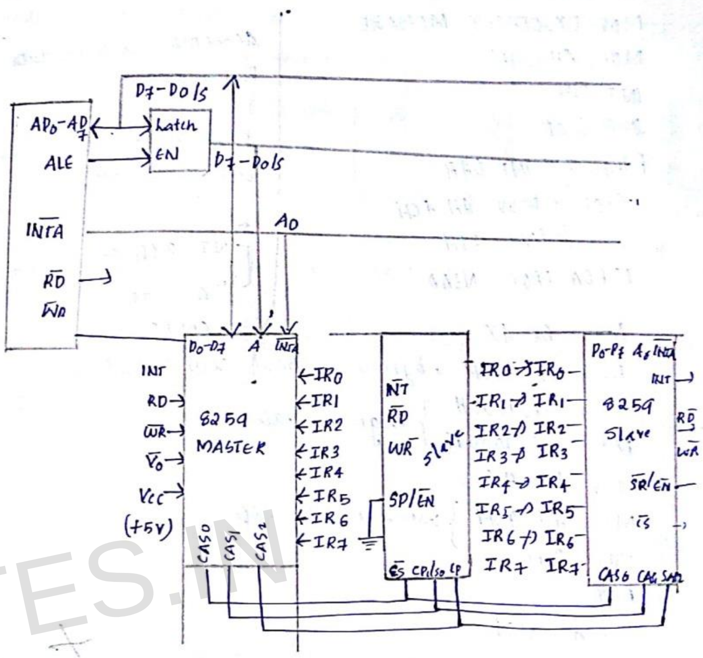
Initialization.

Initialize every 8259
Vector number
Slave
Slave ID

- 64 interrupt can handle we use cascaded interrupt
- Each interrupt have different vector number.
- Vector no: calculate the slave.

Important Application level

- Master connect → Vcc
- Slave connect → ground
- SP / EN
- 3 cascade uses



FILENAME DB "

Msg db file not created successfully

INTERRUPT PROGRAMMING

```
START: MOV AX; CODE
       MOV DS, AX
       MOV DX, OFFSET ISR0A
       MOV AX, 250 AH (Set IVT using function 250AH)
       INT 21H
       MOV DX, OFFSET FILENAME
       MOV AX, DATA
       MOV DS, AX
       MOV CX, 00H
       MOV AH, 3CH ; create a file with filename
                                        'RESULT'
       INT 21H
```

```
JNC FURTHER
MOV DX,OFFSET MESSAGE
MOH AH,09H
INT 21H
JMP STOP
FURTHER: INT 0AH
 STOP : MOV AH,4CH
        INT 21H
 ISROA PROC NEAR

  MOV BX, AX
  MOV CX, 500H , byte count (500)
  MOV DX, 1000H } Segment value.
  MOV AX, 1000H }
  MOV DS, AX
  MOV AH, 40H } write in the file.
  INT 21H
  RET
  ISROA ENDP.
  CODE ENDS
  END START
```

```
Assume SS code DS data
Data segment
   :
Data ends
Code segment
   :
 INT 09h
 ISR 09h
  ENDP
Code ends end
```

Interrupt programming

While programming for any type of interrupt the programmer must, either external / through the program, set the Interrupt Vector table for that type preferable with the CS and IP address of the ISR the method of defining the ISR for S/W as well a H/W interrupt is same.

It shows the execution sequence in case of S/W interrupt. It is assumed that the IVT is initialised suitably to point the ISR

Q write a program to create a file 'RESULT' and store in it 500H bytes from the memory block starting at 1000, if either an interrupt appears at INTR pin with type 0AH or an instruction equivalent to the above interrupt is executed

Q Basic peripherals and their interfacing

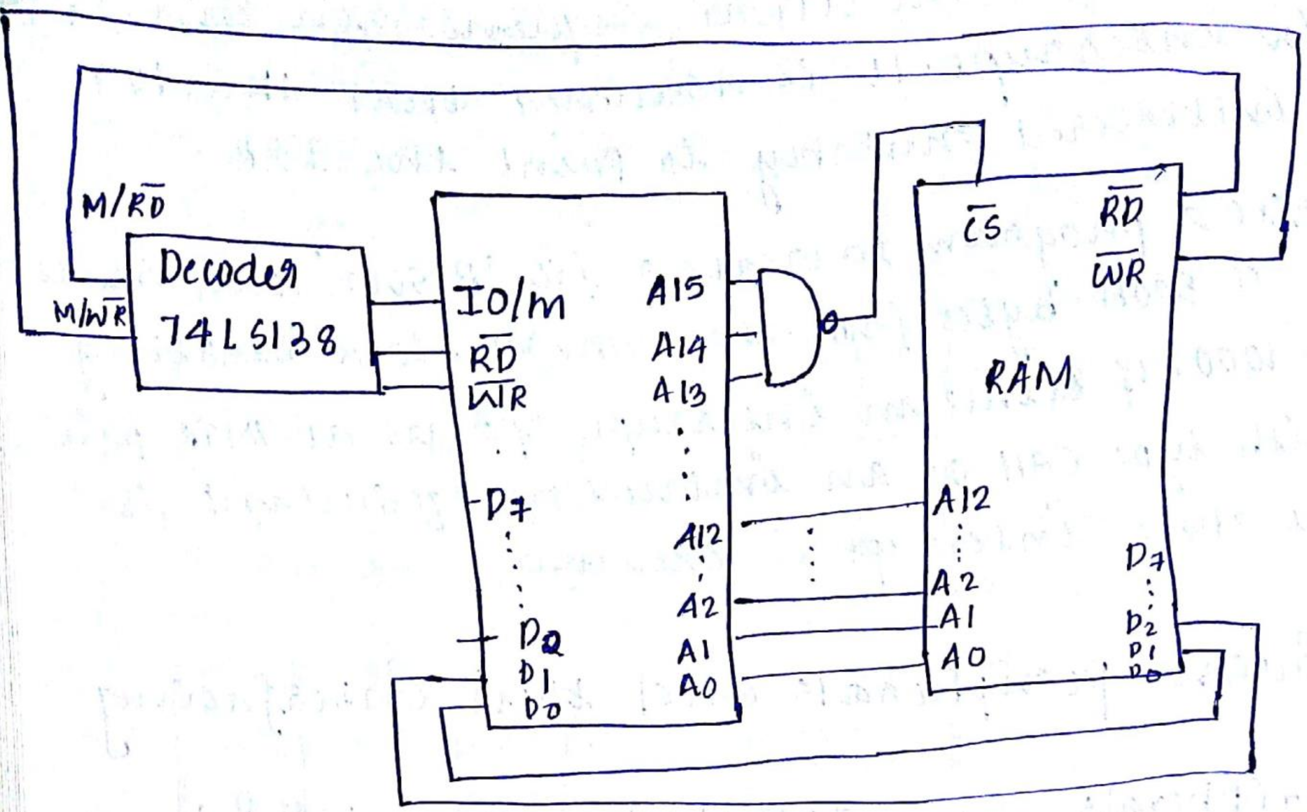
Peripherals	dedicated Peripherals	M/y interfacing
keyboard	using 8259	
display		
memory		
I/o		

→ Memory interfacing

1. Semiconductor m/y interfacing
   - ↳ ROM
   - ↳ RAM

Interface 8K RAM with CPU

1. howmany address lines are required
2. connect the data lines of CPU with m/y
3. decoding logic for address mapping.
4. Proper control signal should be connected b/w CPU and m/y



(8-bit processor)

1. 8K - $2^3 \cdot 2^{10}$ ∴ 13 address lines (A0 - A12)

(If in question $\underset{\substack{\downarrow \\ \text{address}}}{8K} \times \underset{\substack{\downarrow}}{8}$ data lines).

2. Connect data and address lines.

3.

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

we need only 13 pins remaining 3 pins are used for chip selection using different combinations In the above fig we use an NAND GATE

A15 — 1
A14 — 1 o/p = 0 CS becomes high → RAM active.
A13 — 1