Simple Program ⇒

Example-1 Write an assembly language program to add two 8-bit numbess stored in registes A and registes B. Display the result at O/P Port 01H.

Solution:

Memory Address	Mnemonics	Hex Code	Remarks
2000H	MVIA, 20H	3EH	Load 20H in registes A.
2001H		20H	
2002H	MVIB, 30H	06H	Load 30H in registes B.
2003H		30H	
2004H	ADD B	80H	Add Contents o/p registes B with A and store the result in A.
2005H	OUT 01H	D3H	Send the Contents o/p A to O/P Port 01H
2006H		01H	
2007H	HLT	76H	Stop the Program.

[Result : A = 50 , Flags : Sign (S) – 0, Zero (Z) = 0, Parity (P) = 1 , Carry (CY) = 0]

Example-2 Write an assembly language program to add two 8-bit numbers stored in register A and register B. Store the result at the memory location 3000H using STA instruction.

Solution

Memory Address	Mnemonics	Hex Code	Remarks
2000H	MVIA, 20H	3E H	Store 20H in
2001H		20H	register A.
2002H	MVIB, 30H	06H	Store 30H in
2003H		30H	register B.
2004H	ADD B	80H	Add Contents of register B with A and store the result in A.
2005H	STA 3000H	32H	Store the Contents
2006H		00H	of A (i.e. result)
2007H		30H	at the memory location 3000H.
2008H	HLT	76H	Stop the Program.

In this program the result will be stored at memory location 3000H by STA instruction.

Example-3 Write an assembly language program to subtract an 8-bit number stored in register B from an 8-bit number stored in register A. Store the result at memory location 4050H. Also write the expected result and status of flag register.

Solution

Memory Address	Mnemonics	Hex Code	Remarks
2000H	MVI A, 25H	3EH	Store 25H in
2001H		20H	register A.
2002H	MVI B, 12H	06H	Store 12H in
2003H		12H	register B.
2004H	SUB B	90H	Subtract the Contents of reg- isters B from A and store the result in A.
2005H	STA 4050H	32H	Store the result
2006H		50H	at memory location
2007H		40H	4050H.
2008H	HLT	76H	Stop the program.

[Result: A: 13H (0001 0011), flags: Sign (S) = 0, Zero (Z) = 0, Parity (P) = 0, Carry (CY) = 0]

Example-4 If 67H is stored in Accumulator (A) and 78H is stored in register B. Perform the logical AND operation between the above two 8-bit numbers. Store the result in memory location 3050H. Also write the expected result and status of flag register.

Solution

Memory Address	Mnemonics	Hex. Code	Remarks
2000H	MVI A, 67H	3EH	Store 67H in
2001H		67H	register A.
2002H	MVI B, 78H	06H	Store 78H in
2003H		78H	register B.
2004H	ANA B	A0H	Bit-wise AND the Contents of register B with A and store the result in A.
2005H	STA 3050H	32H	Store the result
2006H		50H	in memory location
2007H		30H	3050H.
2008H	HLT	76H	Stop the program.

[Result : A : 60 H, flags : Sign (S) = 0, Zero (Z) = 0, Parity (P) = 1, Carry (CY) = 0]

Example-5 What are the applications of Rotate inst
Wriate a progsam using rotate instructio
to multiply the data stored in accumulato

Solution

Memory Address	Mnemonics	Hex Code	Remarks
2000H	MVI A, 25H	3EH	Store 37H i register A
2001H		25H	
2002H	RLC	07H	Rotate Accumuk Left
2003H	HLT	76H	Stop the progra

[Result: A: 4AH, Flags: CY = 0]

Example-6 What is the difference between SUB an CMP instruction. Store 25H in A and 12H B. Use CMP B instruction and check the of A & B after the execution of CMP instruction

Solution

Memory Address	Mnemonics	Hex Code	Remarks
2000H	MVI A, 25H	3EH	Store 25H in register A.
2001H		25H	
2002H	MVI B, 12H	06H	Store 12H in register B.
2003H		12H	
2004H	CMP B		Add contents of register B with A and store the result in A
2005H	HLT	76H	Stope the program

[Result: A: 25H, B = 12H, Flags: S = 0, Z = 0, P = 0, CY = 0]

Qu: Add two 8-bit number stored in memory location 2050H and 2051H and store the result at memory location 2052H.

Solution

Memory Address	Mnemonics	Hex Code	Remarks
2000H	LXI H,	21H	Load memory address
2001H	2050H	50H	of the first number
2002H		20H	in HL register pair
2003H	MOV A, M	7EH	Copy first number into A.
2004H	INX H	23H	Increment the contents of HL pair. HL pair now pointing to second number
2005H	ADD M	86	Add first & second number
2006H	STA 2052H	32H	Store the result at
2007H		52H	memory location 2052H
2008H		20H	
2009H	HLT	76H	Stop the program

Suppose data stored at

2050H - 23H

2051H - 44H

The result after execution of the program stored at 2052 will be 67H

Example Add two 16-bit number 257DH and 82A1H Using DAD instruction. Store the result in memory

Memory Address	Mnemonics	Hex Code	Remarks
2000H 2001H 2002H	LXI B, 2570H	01H 70H 25H	Load first number in BC register pair
2003H 2004H 2005H	LXI H, 32A1H	21H A1H 32H	Load second number in HL register pair
2006H	DAD B	09H	Add contents of BC with HL. Result is Stored in HL
2007H	MOV A,H	7CH	Copy higher order byte of the result into A.
2008H 2009H 200AH	STA 3050H	32H 50H 30H	Store higher order byte in memory location 3050H
200BH	MOV A,L	7DH	Copy lower order byte of the result into A
200CH 200DH 200EH	STA 3051H	32H 51H 30H	Store lower order byte in memory location 3051H
200FH	HLT	76H	Stop the program.

[Result: HL = 5812H]

Q→ write an assembly language program to performed the following ~~inst.~~ oper. after execution

1→ A=?

2→ F=?

3→ PSW=?

4→ length of program

5→ No. of M/C's

6→ Total execution time if $f_{CLK} = 3MHz$

7→ M/M representation of program if its starts at 9000H

8→ no. of times Read control signal activated for the execution of program.

~~MOV~~

9000	3E	76	MVI A, 76H
9002	43	FR	MVI B, CDH
9004	90		SUB B
9005	57		MOV D, A
9006	76		HLT

A ← 76H
B ← CDH
A − B → (D)
Stop

$10\ 10$
76
CD
A9

CY 1

10101001

S Z X AC X P X CY
1 0 0 0 0 1 0 1

22
13
9

16
6
22
12
10

i→ A → A9

ii→ 85H

3→ A985 | A | PSW |

4→ 7 byte

5→ ~~28~~ M/C

6→ ~~267 μs~~ 9μs

7→

9000	XX
9001	76
9002	YY
9003	CD
9004	ZZ
9005	WW
9006	**

content of D → 85H

f = 3MHz
T = 0.33 μS
T = 0.33 × ~~8~~ 27

Total execution Time = CLK period × T-states × Count Value

$= \frac{1}{3 \times 10^6} \times 27 \times 1$

How many time program is executed

8→ $\overline{RD}$

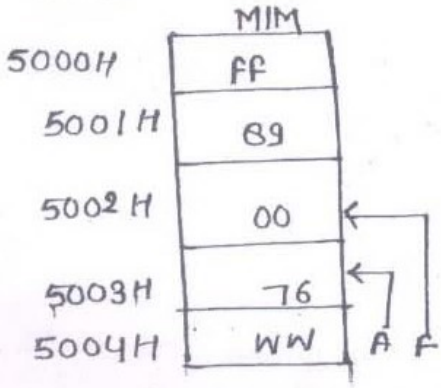
a→ ?
b→ 5
c→ 7 ✓
d→ none

9→ No. of time ALE is activated.

9→ ALE

a→ 5
b→ 7 ✓
c→ 8
d→ none

Q1→ write an ALP for 8085 to perform X-OR operation between first two memory locations content & store the content of Flag register & accumulator in the next two memory locations.



XRA R, XRA M, XRI 8 bit data.

```
LXI H, 5000H
MOV A, M
INX H
XRA M
LXI SP 5004H
PUSH PSW
HLT
```

or

```
LDA 5000H
LXI H, 5001H
XRA M
LXI SP, 5004
PUSH PSW
HLT
```

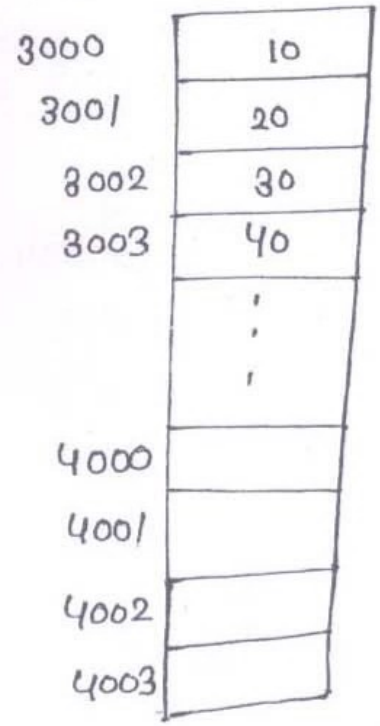
Q3 write an ALP to access a data byte from input port 40H

→ Compliment It
→ Rotate it left for 5 time
→ Transfer the resultant value to port add- 50H

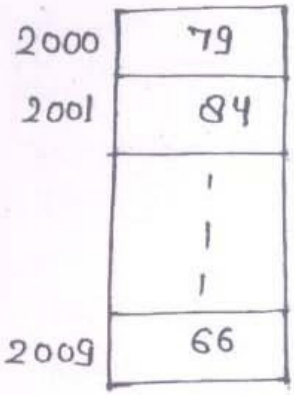
```
    IN 40H
    MVI C 05H
    CMR
Start RLC
    DCR C
    JNZ Start
    OUT 50H
    HLT
```

IAS 10M
Q4 → Write an ALP to transfer 4 bytes of data starting from 3000H to 4000H.

```
    LXI H, 3000H
    LXI D, 4000H
    MVI C, 04H
Rept: MOV A, M
    STAX D
    INX H
    INX D
    DCR C
    JNZ Rept
    HLT
```



Q→ Write an ALP to find no. of even and odd no. from 10 bytes of data starting at 2000H. Store the even count in Register E & odd count in Register D.

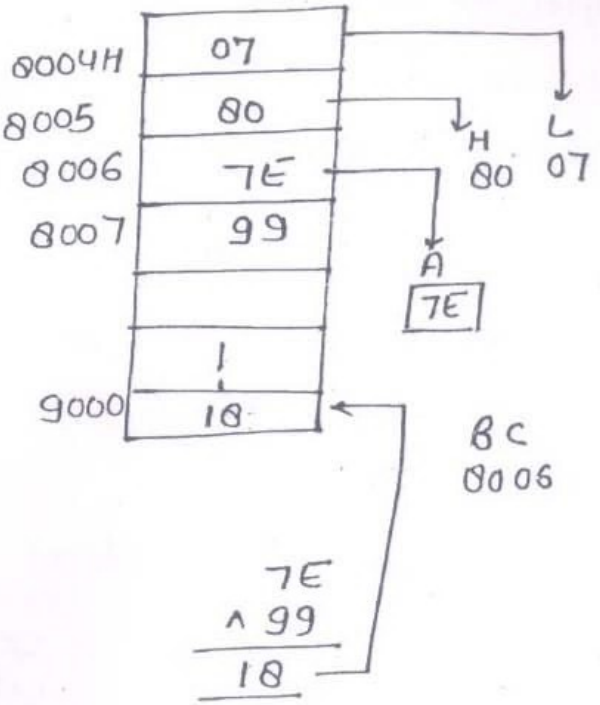



```
LXI D, 0000H
LXI B, 2000H
MVI L, 0AH
L1: LDAX B
    RRC
    JC odd
Even. INR E
    JMP NEXT
Odd: INR D
NEXT INX B
    DCR L
    JNZ L1
    HLT
```

```
LXI D 0000H
LXI H 2000H
MVI B 0AH
L1  MOV A, M
    RRC
    JC : ODD
Even, INR E
    JMP NEXT
ODD ; INR D
NEXT INX H
    DCR B
    JNZ L1
    HLT
```

Q→ Find the content of SP and data at top of Stack after execution

```
LXI SP, E00FH
LHLD 8004H
LXI B , 8006H
LDAX B
ANA M
STA 9000H
PUSH PSW
HLT
```

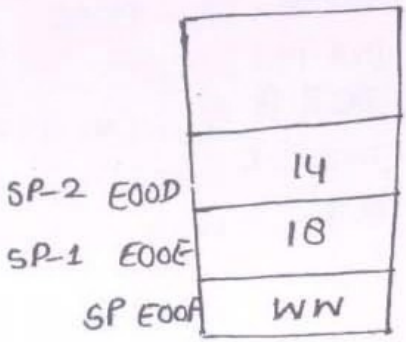



```
01111110
10011001
--------
00011000
--------
```

S	Z	X	AC	X	P	X	CY
0	0	0	1	0	1	0	0

14H

PSW
1814H



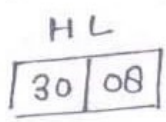
SP = E00D
data = 14H

Q→ Find the value of SP and data present at top of the stack after execution of the program.

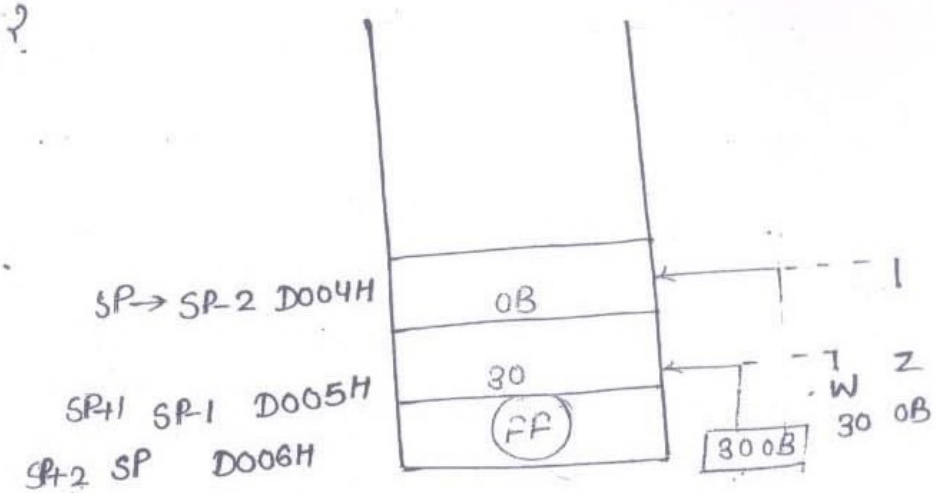
```
3000H: LXI SP, D006H
3003H: LXI H, 3008H
3006H: PCHL
3007H: MOV A, L
3008H: CALL R2
300BH: JMP Quit
300EH: XRA A
300FH: RZ
3010H: HLT
```

R2

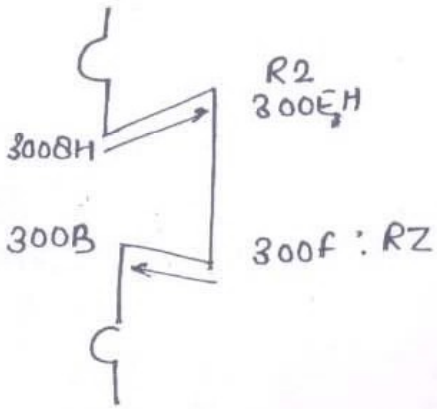
Quit:



A=00H



SP→ D006H
Data→ ~~0~~ FFH



Q→ i→ Find the content of SP,

```
LXI SP, FFFFH
LXI H, 1234H
LXI D, 5678H
PUSH H
PUSH D
POP H
POP D
HLT
```

Content of SP = FFFFH

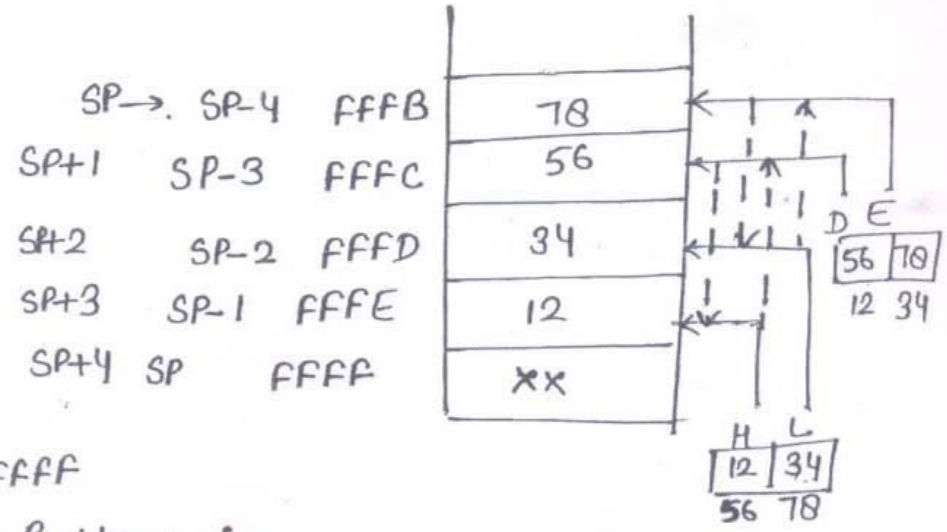
Q→ One of the following True after execution

a→ DAD D

b→ XTHL

c→ XCHG ✓

d→ None



SP → FFFF

Timers & Counter Problems :-

Total execution Time = Clock period x No. of T states x Count value.

i→ Timer Problems

Q→ Find the total execution time for a program in 8085 if it consumes 600 T states and a crystal freqⁿ is 6MHz

$$= \frac{1}{3 \times 10^6} \times 600 = 200\,\mu s$$

$$f_{CLK} = \frac{f_{crystal}}{2} = 3MHz$$

Q→ Find the total execution time when CLK freqⁿ is 5MHz

10T LXI B, AA55H → 3 M/C → 10T

4T MOV A, C → 4T

4T ANA B → 4T

7/10 JPE S1 = 1 → 10

7 MVI A, FFH

7 ORI 00H

4 MOV B, A

5 S1 HLT → 5

A = 55

B = AA

0101 0101

1010 1010

00000000

$$\frac{2\ 3+10}{5} = \frac{33}{5} = 6.6\,\mu s$$

Q. Find the total execution time if operating freq^{n} is 3MHz

```
      MVI C, 03H   7
R1    NOP          4
      DCR C        4
      JNZ R1       7/10
      HLT          5
```

$$\frac{1}{8} \times 30 \left(\cancel{3}\right)^2 =$$

$$TET = T_{OL} + T_{WL}$$

$$T_{OL} = \frac{1}{3\times10^6} \times (7+5)\times 1 = 4\mu s$$

$$T_{WL} = T_{CS} + T_{CNS}$$

$$= \frac{1}{3\times10^6} [10\times 2 + 15\times 1]$$

$$= 17$$

$$T.E.T \ \cancel{IEE} = 4+17 = 21\mu s$$

Q. T.E.T = ?

f_{CLK} = 5MHz

```
      MVI B, 05H   7        B = 05
      MVI A, 00H   7        A = 00
Rpt   ADD B        4        A = 05
      DCR B        4
      JNZ Rpt      7/10      5
      ADI 04       7
      HLT          5
```

$$T_{OL} = \frac{1}{5\times10^6} [7+7+7+5] \times 1 = \frac{26}{5} = 5.2\mu s.$$

$$T_{WL} = \frac{1}{5\times10^6} [4\times 18 + 15\times 1] = 17.4\ \mu s$$

$$T.E.T = T_{OL} + T_{WL}$$

$$= 5.2 + 17.4$$

$$= 22.6\ \mu s$$

A	B
00	05

~~-05~~
~~05~~

00
05
A → 05H

	A	B
1 →	05	04
	04	
	09	
2 →	09	B = 03
	03	
	0C	
3 →	0C	B = 02
	02	
	0E	
4 →	0E	B = 01
	01	
	0F	
5 →	0F	B = 00
	04	
A	13	

True

False

A = ?
a → 19H
b → 09H
c → 13H ✓
d → 16H

Counter Problems

```
      MVI B, XX  → 4+3 = 7
L1 →  LDA 5900H  → 13
      STA 5900   → 13
      DCR B      → 4
      JNZ L1     → 7/10
      NOP        → 4
      HLT        → 5
```

n-1 → True
n → To

Q → T.E.T = 500 μS
f_{CLK} = 5MHz

what should be ~~that~~ value of count in Register B is loaded such that T.E.T = 500 μS.

$$T_{OL} = \frac{1}{5\times10^6} \times [16] \times 1 = 3.2\ \mu S$$

$$T_{WL} = \frac{1}{5\times10^6} [xx \times 40 + 1 \times 37]$$

$$500\times10^{-6} = \frac{(40x + 37)}{5\times10^6} + \frac{16}{5\times10^6}$$

$500\times10^{-6} \times 5\times10^{6} = 40x + 37 + 16$

$2500 = 53 + 40x$

$40x = 2447$

$40(m-1) = 2447$

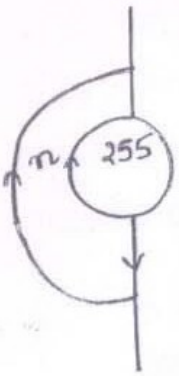
$40m - 40 = 2447$

$40m = 2487$

$m = 62.175 \approx (62)_{10} = 3E\ H$

$\boxed{x = m-1}$

Q→ Find the count value in Register B such that the total execution time is 100mS. when the processor operates with 2MHz clk.



	MVI B, XX	7	→ 7×1=7
L2:	MVI C, FF	7	→ 7×m
L1:	DCR C	4	→ 4×255×m
	JNZ L1	7/10	→ m[10×254 + 1×7]
	DCR B	4	→ 4×m
	JNZ L2	7/10	→ 10(m-1) + 7×1
	HLT	5	→ 5×1 = 5

L1 – L2 → 255 Times

True → 254

false → 1

L2 → m Time

True → m-1

false → 1

$= \frac{1}{2\times10^{6}} [7+5] = 6\ \mu S$

$T.ET = T_{OL} + T_{WL}$

$T_{WL} = T_{ET} - T_{OL}$

$= 100\ mS - 6\ \mu S$

$= 100 \times 10^{-3} \times 10^{-3} \times 10^{3}\ 6\ \mu S$

$= 100000\ \mu S - 6\ \mu S$

$= 99994\ \mu S$

~~$t_1 = \frac{1}{2\times10^6}$ [254 × 14 + 11 × 4]~~

~~t_1 = 1783.5 μs~~

~~t_2~~

$$n = (55.7)_{10}$$
$$\approx (56)_{10}$$
$$= 38H$$

$$T.E.T = 7 + 7n + 1020n + 2540n + 7n + 4n + 10n - 10 + 7 + 5$$

$$100ms = \frac{1}{2\times10^6}(9 + 3588n)$$

$$200 \times 10^3 = 9 + 3588n$$

$$n = 55.7 \approx (56)_{10}$$

$$n = (38)_H$$

Q→ How many times loop will execute

a→2
b→1
C→ infinite ✓
d→ none

```
        if Z=0
        LXI D, 0002H
L1 :    DCX D
        JNZ L1  Z=0?
        HLT
```

DE→ 0002
after decrement
0001

In DCX Rp → no Flags are affected.

Q→ XRA A ; Z=1
LXI D, 0002H
L2 : DCX D
JNZ L2 0?
HLT

a→ 2
b→1 ✓
c→3
d→∞

Q3→ LXI D, 0002H
0
L3 : DCX D
?
JNZ L3
HLT

Find the set of instruction that must be included before jump such that loop rotates for n no. of times according to count value load in DE register pair

a→ MOV A, E
ANA D

b→ MOV A, E
ORA D

c→ MOV A, E
XRA D

d→ None

i→
D	E
00	02
00	01

A
01
^ 00
00 : Z=1

ii→
DE
01 02 ⇒ 258 Time

A
01
01
00 Z=1

loop 1 time

iii→
DE
0002
0001
0000

A
01
V 00
01

A
00
V 00
00 : Z=1

Interrupts

It is an internal or external signal which may disturb or alter the sequence of execution of the processor. Interrupts are classified as-

1 → Hardware & Software

2 → Maskable & Non Maskable

3 → Vectored & Non-Vectored

Hardware Interrupts

There are 5 hardware interrupts according to priority
TRAP, RST 7.5, RST 6.5, RST 5.5, INTR.

- They must be connected to an application physically.

Software Interrupts

There are 8 software interrupts which can be used as instructions or along with INTR. eg

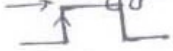
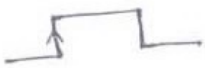
	OPCODE
RST 0	XX
RST 1	YY
⋮	⋮
RST 7	ZZ

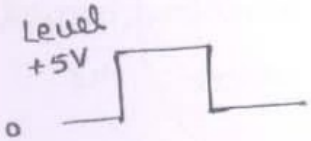
Maskable Interrupt →
Interrupt which can be neglected, ignored or avoided even they are triggered are maskable.

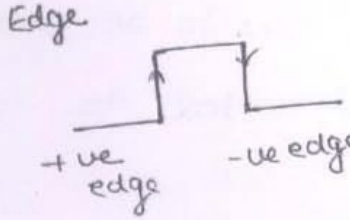
Non-Maskable Interrupt →
- Interrupt which cannot ignored or avoided when they are triggred.
- TRAP is the only non-Maskable interrupt in 8085. It must be connected to highly prioritised event in practical application

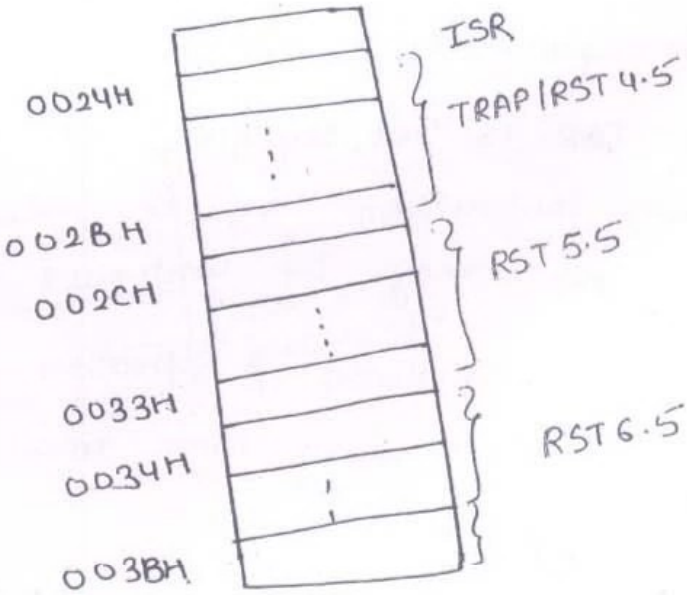
Vectored Interrupt
- Interrupts which have specific address location.

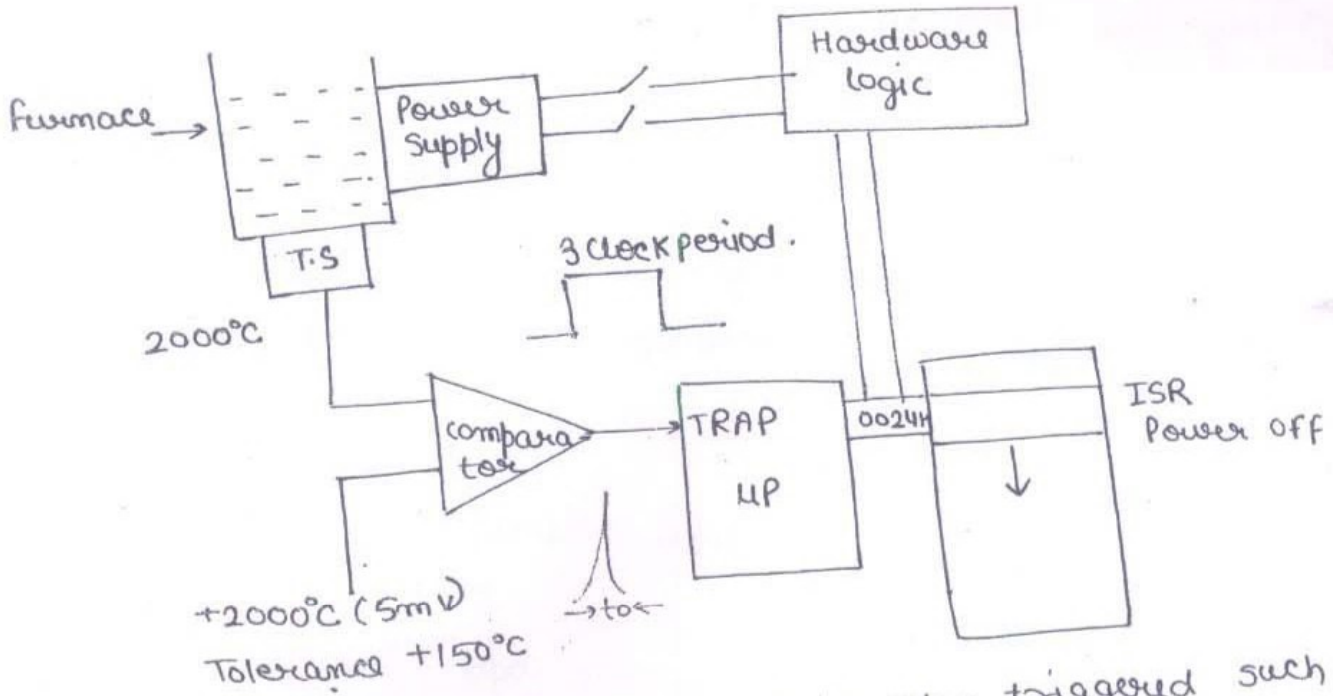
Non-Vectored
- Interrupt which don't have specific add. location.
  INTR is the only non-vectored interrupt in 8085.

Priority	Interrupt	Type of Triggering	Add. Location
1	TRAP		0024H
2	RST 7.5		003CH
3	RST 6.5		0034H
4	RST 5.5		002CH
5	INTR		—

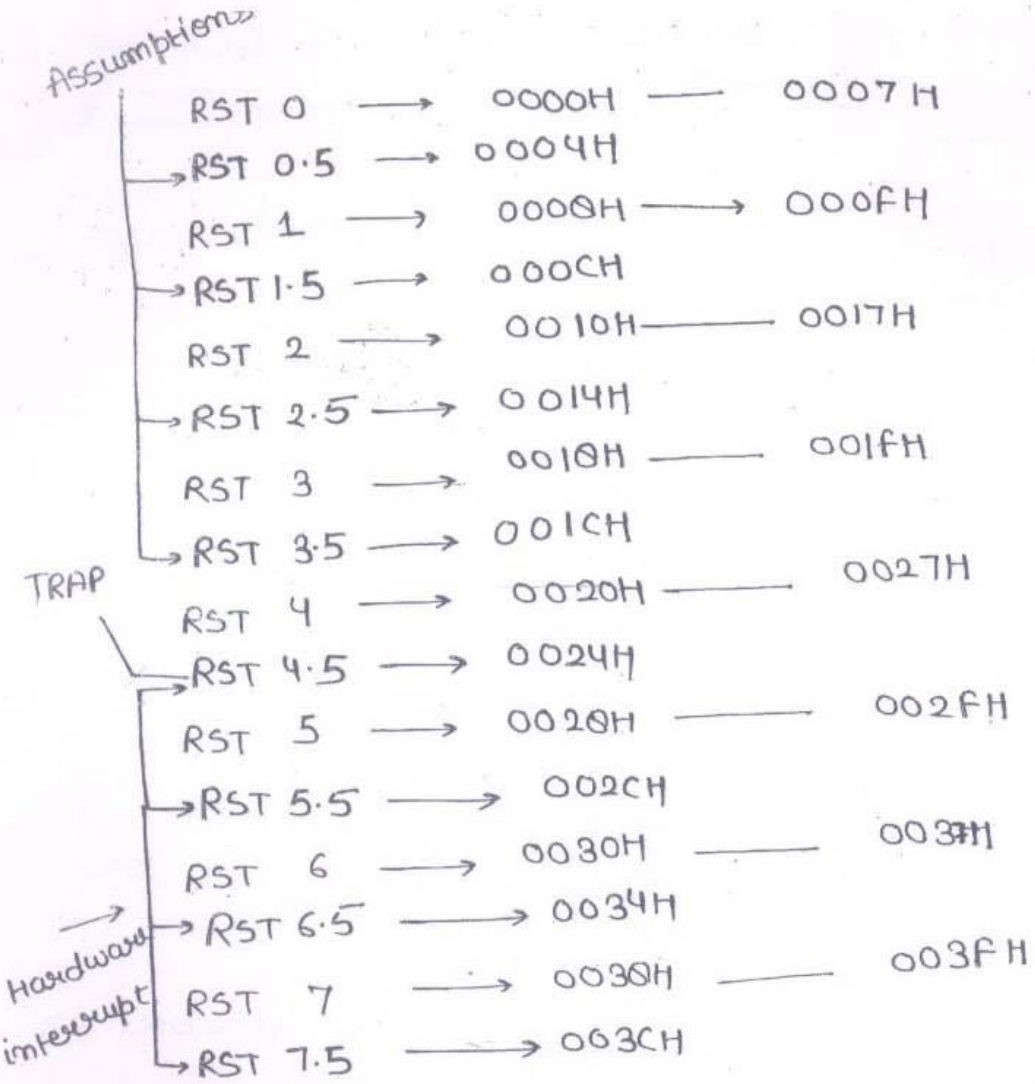


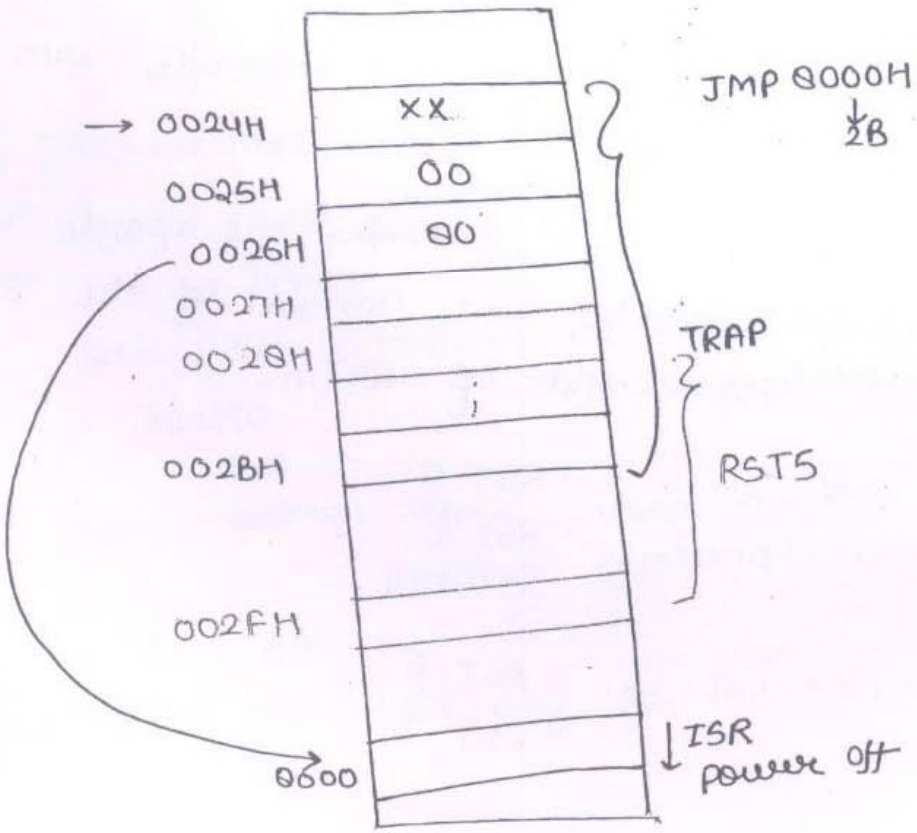




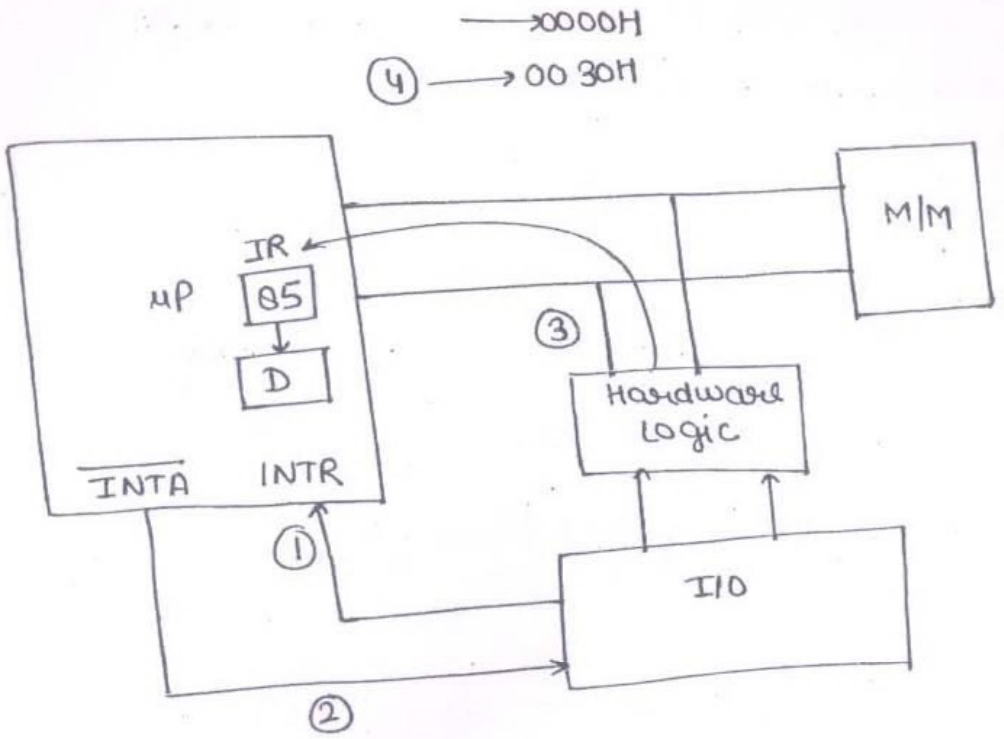


- Trap is both edge & level trigger. It is edge triggered such that it may be responded quickly.
- It is level triggered in order to differentiate b/w signal from original practical application & error signal due to noise.
- The signal on TRAP pin must be high for 3 clock periods in order to avoid error signal due to noise.
- every vectored interrupt is assigned 8 bytes of memory to store the corresponding program that must be executed in response to an interrupt which is known as interrupt service routine.
- If ISR is not sufficient in 8 bytes an unconditional Jump instruction can be included such that the program may be continued further.
- when both hardware & software interrupts are to be used proper programming Logic must be implemented to avoid errors.


Assumptions
RST 0 → 0000H — 0007H
RST 0.5 → 0004H
RST 1 → 0008H → 000FH
RST 1.5 → 000CH
RST 2 → 0010H — 0017H
RST 2.5 → 0014H
RST 3 → 0018H — 001FH
RST 3.5 → 001CH
TRAP
RST 4 → 0020H — 0027H
RST 4.5 → 0024H
RST 5 → 0028H — 002FH
RST 5.5 → 002CH
RST 6 → 0030H — 0037H
Hardware interrupt
RST 6.5 → 0034H
RST 7 → 0038H — 003FH
RST 7.5 → 003CH


0024H
XX
0025H
00
0026H
80
0027H
0028H
002BH
002FH
0800
JMP 8000H
2B
TRAP
RST5
ISR
power off

INTR



- The programmer must select one of the software interrupt vector address to store the corresponding ISR for the I/O.
- An external hardware logic can be design to produce or generate the opcode of selected RST n , n→ 0 to 7.
- When the processor receives INTR it responds with $\overline{INTA}$ and waits for the response from I/O.
- The external hardware logic provides the opcode which is exist into IR , decoded and control of the program is transfer to vector add. of RSTn.

	OPCODE
RST 0	XX
RST 1	YY
:	
:	
RST 6	05
RST 7	ZZ

- As there is no specific address for INTR it is known as non vectored interrupt.

NOTE→

- $\overline{INTA}$ is only required only for INTR not for vectored interrupt.
- Vector Address Calculation for Interrupts

RST n ; n → 0 → 7

$n \times 8 = (X)_8 = (Y)_{16}$

eg→ RST 5

$5 \times 8 = (40)_{10}$ 0028H

16	40	8
	2	

RST 7.5

$7.5 \times 8 = (60)_{10}$

16	60	C
	3	

003CH

Enable Instruction Related to interrupts

EI → enable interrupts

DI → Disable "

SIM → Set interrupt mask → used to mask the interrupts (or) make them available

RIM → Read interrupt mask → used to know status of pending interrupts

SIM & RIM are valid only for RST 7.5, RST 6.5 & RST 5.5

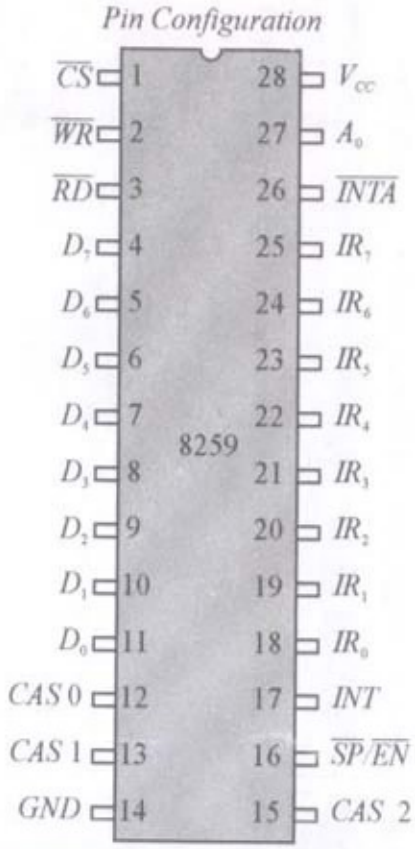
PROGRAMMABLE INTERRUPT CONTROLLER (PIC) - 8259

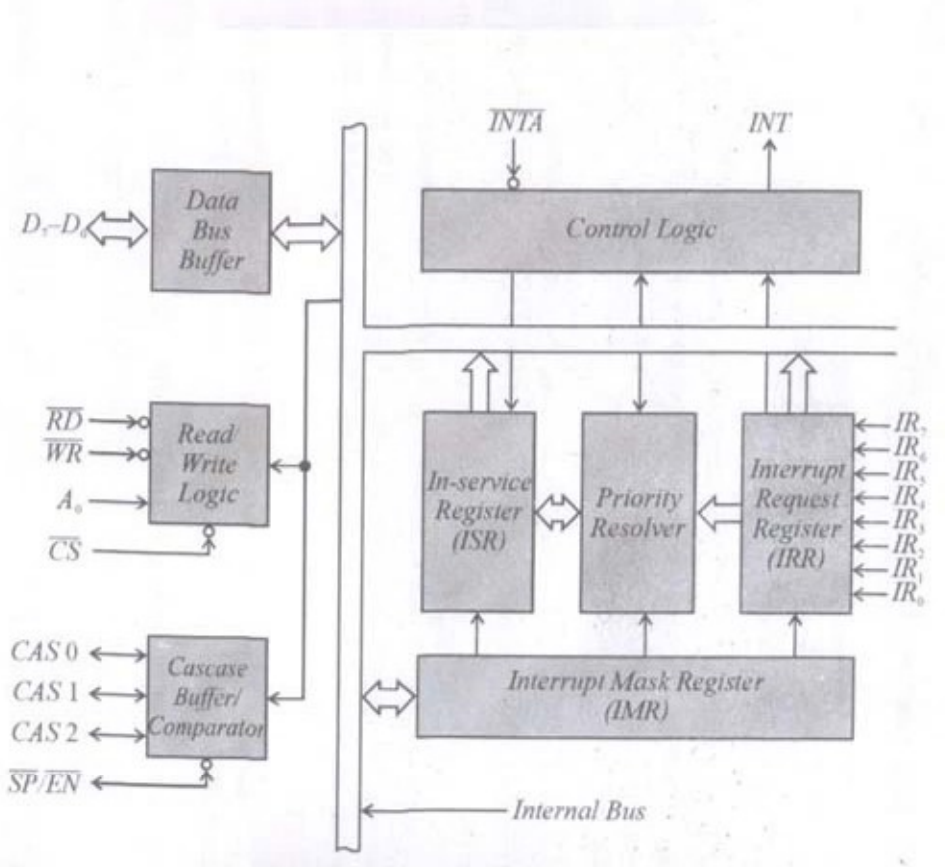
Apart from the various software (RST0 to RST7) & hardware (TRAP, RST 7.5,6.5,5.5 and INTR) linterrupts of 8085, the 8259 IC can be used to provide additional interrupts if required. The 8259 has following features:

1. It provides 8 additional interrupts by connecting it to INTR pin of 8085 microprocessor. Thus we have total 12 interrupts lines by using one 8259 (8 new and 4 earlier interrupt lines (TRAP, RST 7.5, RST 6.5, RST 5.5), with the loss of INTR line).
2. Interrupt levels can be extended to 64 lines, by cascading additional 8259s. Maximum nine 8259s may be connected in cascade (one as Master and eight as Slaves)
3. It vectors an interrupt request anywhere in the Read/Write memory. But, these interrupts are spaced at the interval of either 4 or 8 locations. The drawback of internal interrupts of 8085 is that they are vectored to the Read Only Memory (00 page).
4. It provides eight levels of interrupt priorities in different modes, such as, Fully Nested Mode, Automatic Rotation Mode and Specific Rotation Mode.
5. It can mask each interrupt request individually.
6. It can read the status of pending interrupts, in-service interrupts and masked interrupts.
7. It can be configured for level-triggered or the edge-triggered interrupt request.
8. It is designed to work with 8085, 8086 and 8088 microprocessors.

PIN DIAGRAM & BLOCK DIAGRAM OF 8259

It is 28 pin IC chip, whose pin diagram and block diagram are as shown:





BLOCK DIAGRAM

8259 PIN FUNCTIONS

The functions of various pins and their signals are explained below:

D_7-D_0 – Connected to data bus.

A_0 – Command select Address. It is connected to A_0 address line of the microprocessor.

IR_0-IR_7 – Interrupt Request Inputs. The interrupt request signal may be connected to any of these pins.

INT (Interrupt) – It is an active high output signal. It goes high when any interrupt is recognised by 8259. This is connected to INTR pin of the microprocessor.

$\overline{\text{INTA}}$ (Interrupt Acknowledge)– It is an active low input signal. This is connected to $\overline{\text{INTA}}$ pin of the microprocessor

CAS2-CAS0 (Cascade lines) – These lines are used to connect more then one 8259s in cascade mode.

$\overline{\text{SP}}$ / $\overline{\text{EN}}$ (Slave Program/Enable Buffer) – It is used to define the 8259 as master/ slave. When single 8259 is used, it operates as a master. Then $\overline{\text{SP}}$ / $\overline{\text{EN}}$ pin is connected to +5 V (logic 1). When more then one 8259s are used then only one 8259 may be a master. The remaining 8259s must operate as slave, with $\overline{\text{SP}}$ / $\overline{\text{EN}}$ connected to ground (logic 0).

$\overline{\text{RD}}$ (Read), $\overline{\text{WR}}$ (Write) and $\overline{\text{CS}}$ (Chip Select) pins have their usual functions.

Vcc – Connected to +5 V; **GND** – connected to 0V.

REGISTERS OF 8259

The 8259A has three important registers, IRR, ISR and IMR.

IRR (Interrupt Request Register) – It stores various interrupt requests, received at pins IR_0-IR_7.

ISR (In-Service Registers) – It stores the information about the interrupt request being served.

IMR (Interrupt Mask Register) – It stores the masking information of masked/ unmasked interrupts.

Functional Description

The 8259 has eight interrupt request inputs, IR0 to IR7. The 8259 uses its INT output to interrupt the 8085 via INTR pin. The 8259 receives interrupt acknowledge pulses ($\overline{\text{INTA}}$) from the 8085 microprocessor at its input. Vector address used by the 8085 to transfer control to the service subroutine of the interrupting device, is provided by the 8259 on the data bus. The 8259 is a programmable device that must be initialized by command words sent by the programmer to its control register. After initializing the 8259, mode of operation can be changed by operation command words from the programmer.

The descriptions of various blocks in the given block diagram are,

Data bus buffer

This tristate, bidirectional 8-bit buffer is used to interface the 8259 to the system data bus. Control words and status information are transferred through the data bus buffer.

Read/Write & control logic

The function of this block is to accept OUTPUT commands from the CPU. It contains the initialization command word (ICW) register and operation command word (OCW) register which store the various control formats for device operation. This function block also allows the status of 8259 to be transferred to the data bus.

Interrupt Request Register (IRR)

IRR stores all the interrupt inputs that are requesting service. Basically, it keeps track of which interrupt inputs are asking for service. If an interrupt input is unmasked, and has an interrupt signal on it, then the corresponding bit in the IRR will be set.

Interrupt Mask Register (IMR)

The IMR is used to disable (Mask) or enable (Unmask) individual interrupt inputs. Each bit in this register corresponds to the interrupt input with the same number. The IMR operates on the IRR. Masking of higher priority input will not affect the interrupt request lines of lower priority. To unmask any interrupt the corresponding bit is set '0'.

In Service Register (ISR)

The In Service Register keeps track of which interrupt inputs are currently being serviced. For each input that is currently being serviced the corresponding bit will be set in the in service register. Each of these 3-registers can be read as status registers.

Priority Resolver

This logic block determines the priorities of the set in the IRR. The highest priority is selected and strobed into the corresponding bit of the ISR during $\overline{\text{INTA}}$ pulse.

Cascade Buffer/Comparator

This function block stores and compare the IDs of all 8259's connected. The associated 3-I/O pins (CAS0-CAS2) are outputs when 8259 is used a master. Master and are inputs when 8259 is used as a slave. As a master, the 8259 sends the ID of the interrupting slave device onto the CAS2-CAS0. The slave thus selected will send its pre-programmed subroutine address on to the data bus during the next one or two successive $\overline{\text{INTA}}$ pulses.