Registers

User accessable Register (General purpose register)
- → B
- → C
- → D
- → E
- → H
- → L

} 8bit

Special purpose Register

User Access
- 8-bit → Accumulator (A)
- 8-bit → Status regi (flag reg.)
- → Stack pointer (SP)
- → Program Counter

} 16 bit

User not Access
- 8-bit → temporary reg.
- 8-bit → interrupt Reg.
- → Latches → 16 bit

Note → in 8085, 8, 8bit register and two 16 bit register (user access)

Accumulator – it is 8-bit special type of Register.

→ when Arithmatic & Logic operation b/w the two number, one number always taken from Accumulator and result of Arithmatic and logic operation will store in Accumulator

Status Register (flag reg. F)

→ it is 8-bit special type register.

→ in 8085 five flags are define at the diffeunt bits of this register.

D7	D6	D5	D4	D3	D2	D1	D0
S	Z	X	AC	X	P	X	CY

S → Sign flag
Z → Zero flag
X → don't care
AC → Auxiliary carry flag
P → parity flag
CY → Carry flag

Note → Status of flag affect according the result of arithmatic and logic operation.

Sign flag (S) →

if in the result of Arith. & logic operation D7 bit or MSB is zero
- D7 = 0 (+ve) → S = 0 (Reset) (no is +ve)
- D7 = 1 (-ve) → S = 1 (Set) (no is -ve)

If in the result of Arith & Logic operation all bits Zero then Z = 1 (Set)

If any bit is one Z = 0 (Reset)

Parity flag →

If in the result of Arith & Logic operation no. of one's are even then P = 1 (Set)

If no. of ~~1's~~ are odd then P = 0 (Reset)

- 8085 is odd parity based system.

:- Result of Arith & Logic

0 0 0 0 0 0 0 0

S = 0
Z = 1
P = 1

Carry flag (CY) :-

~~If~~ in the arith & Logic operation Carry pass from D_7 or MSB then

CY = 1 (Set)

Otherwise CY = 0 (Reset)

T→ In subtraction operation Carry flag can work also Borrow flag.

→ If Borrow is taken at D_7 bit then

CY = 1

otherwise CY = 0

Ex:
```
  1 0 1 1 0 0 1 0
  0 1 1 1 0 0 1 0
1 0 0 1 0 0 1 0 0
```
CY = 1 (Set)

uxillary Carry flag :-

If Carry pass from lower nibble to upper nibble or $D_3 \rightarrow D_4$

AC = 1 (Set)

otherwise AC = 0 (Reset)

Eg:

upper				lower nibble			
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0

~~For programmer~~ Status of four flag available. CY, Z, S, P.

~~For~~ Flag ~~is~~ available.

Imp → AC flag used internally for the adjustment of content of Accumulator in BCD format by assuming earlier operation was BCD addition operation. (13)
When DAA (Decimal Adjustment of ACC) inst. will execute.

Stack pointer :- (SP)

→ it is 16 bit special type Reg
→ SP hold the add. of top of stack that define inside the Memory.

Programm Counter :- (PC)

→ it is 16 bit special type of Register.
→ Programm counter hold the address of next instruction to be fetch.

$$PC \leftarrow PC+1$$

NT → For the storage of 16 bit data then two register can be combined in specific way-

B C ≡ B
D E ≡ D
H L ≡ H

→ higher byte of 16 bit data always store in higher order register = B, D, H

→ lower order byte of data always store in C, E, L

NT **PSW (Programm status word)**

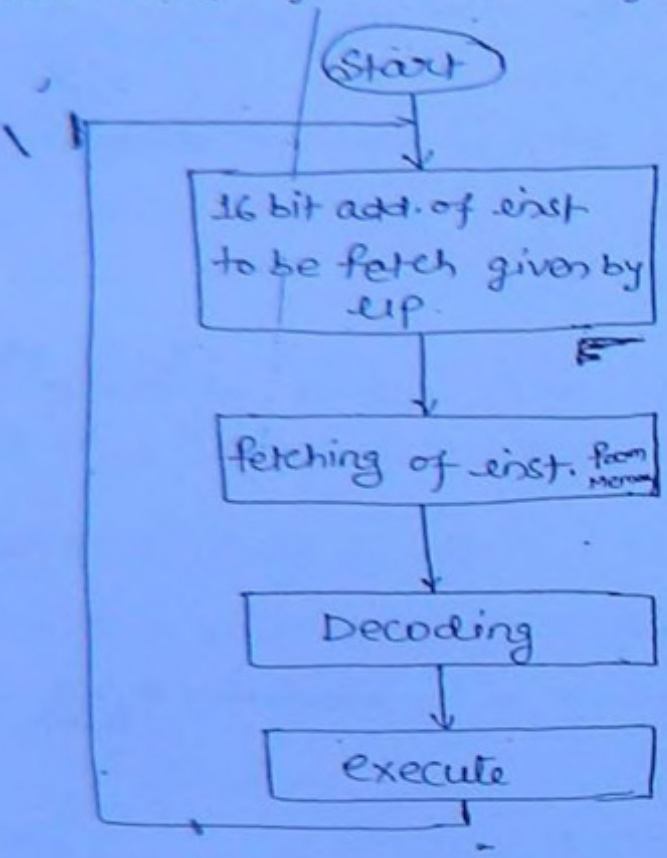
it is also defined 16 bit register in which higher 8-bits are accumulator & lower 8 bit status register

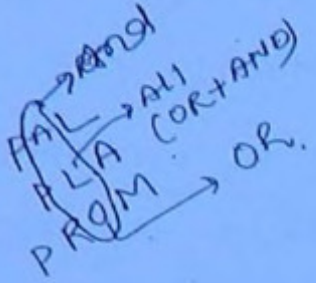
$$PSW = AF$$

A- Accumulator

instructions

(i) flow chart of execution of program





(14)

instruction :-

→ It is command that given to ip to perform a task

→ every instruction has two field.

Opcode → Operational code → Which type of task to be perform

Operand → on which data

→ Lable : opcode operand ; discription

Lable → add of inst.

opcode operand → instn

discription → operation (optional)

→ in 8085, 74 opcodes are define and by using them 256 inst are possible but 246 instructions are available.

MOV B, C } both are different inst.
MOV A, B }

→ Ex. MOV A C = 1 byte (27H)

MVI B, 28H = 2 byte (F0H)

LXI D, 2589H = 3-byte (90H)

1000 : 27
1001 : F0
1002 : 28
1003 : 90
1004 : 89
1005 : 25

(15)

(MOV) program always store in continuous memory location because we use PC ← PC+1

→ no. of memory byte register to feed any inst. that is known as IWS (inst. word size).

⇒ on the basis of IWS, instructions can be classified in three groups.

(1) 1 byte (IWS = 1 byte)
(2) 2 byte (IWS = 2 byte)
(3) 3 byte (IWS = 3 byte).

(i) if in the inst R, Rp (or) no operand then IWS = 1 byte.

R = Register
Rp = Register pair

Ex

MOV A, B
NOP
LDAX B
} IWS = 1 byte.

(ii) if in the inst. 8 bit no. (add. or data) is present IWS = 2 byte

Ex MVI C, 25H = IWS = 2 byte

IN 29H = 2 byte

(iii) if in the inst. 16 bit no. (add. or data) is present then

IWS = 3 byte.

Ex

LXI SP, 2718H = 3 byte = IWS

LDA 5586 = IWS = 3 byte.

NT →

if a 16 bit no stored in memory then higher order byte always at higher order add. (memory add.) lower order byte always at lower order add. (memory add.).

** → in 8085 i/p and o/p port add. of 8 bit

so maximum no. of i/o device = $256 - 0^{8}$ = more no. of o/p device

(at a time) data devices = 256+256 = 512.

instruction
- NAME
- FORMATE
- IWS
- operation
- Add. Mode.
- flag - status
- machine cycle
- T- state.

Group I

8-bit data transfer instruction :-

MOVE instruction.

Rd = destination Register
Rs = source register

Rd } A, B, C, D, E, H, L &
Rs } M

(i) MOV Rd, Rs

ex MOV B, C

IWS = 1 byte

operation - Content of Rs will pass in Rd (copy of)

[Rd] ⟵ [Rs]

[[ ]] ← data
add

Ex

B = 26
C = 59H
MOV B, C
B = 59H
C = 59H

(ii) MVI R, 8 bit data

(move immediately)

IWS = 2 byte

operation - [R] ⟵ 8 bit data { 8 bit data will store in R

R: A B C D E H L & M

Ex MVI B, 29H

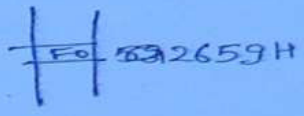
B = 29H

} if last character of opcode I then data otherwise address

M — it is 8-bit user define register - in memory. and address of M is the content of HL pair.

Ex

[H] = 26H
[L] = 59H
MOV A, M
A = F0



(iii) IN 8 bit port add.

IWS ≡ 2 byte

Operation – when this inst. will execute data available at the 8-bit port address will store in Accumulator.

[A] ← [8-bit port Add.]

Ex IN AFH

[A] ≡ BEH

10111110

10101111 port add.
A F
keyboard

'P' – 10111110
B E

(iv) OUT 8 bit port add.

IWS = 2 byte

Operation :– when inst. will execute content of Accumulator will available at 8-bit port add. given in the inst.

Ex:

```
MVI B, 92H
MOV A, B
OUT 12H
```

Note : All above four inst. are data transfer inst. So status of flag will not affect

Group II:-

Machine control instruction:-

(18)

1) NOP no operand

NOP

IWS = 1 byte

Operation:- When this inst. will execute processor will not perform any task.

Note:- this inst. is used for creation of delay

2) HLT No operand.

IWS = 1 byte

Operation:-

When this inst. will execute further increament of program counter will stop.

Note:- it is last inst. of every program.

Ex

1000H MOV BC
1001 MVI A 29H
1003 IN 20H
1005 MOV DE
1006 HLT

PC = 1007

Note:- all above two inst. are machine control inst. and that will not affect status of flag.

Addressing Mode :-

→ Form of add. of data given in the inst is known as add. Mode

① Register add. Mode :-
if add. of data given in the form of Register then Reg. add.

Ex: MOV B, C

(19)

② Direct add. Mode :-
if add of data given directly in the inst.

Ex: IN 25H
OUT 32H

③ immediate add. Mode :-
if data itself given in the inst.

Ex MVI D, 29H } if opcode last character I then immediate add. Mode
Reverse not correct

④ indirect add. Mode :- indirect Reg. add. Mode.
if Content add. of data given in the form of Content of Reg. then indirect add. Mode

Ex: MOV B M.

⑤ implicit add. Mode :-
if add of data not require it is define in opcode only

Ex: NOP, CMA.

Group (III)

8-bit Arithmetic instructions

(i) ADD R

IWS = 1 byte

R → A, B, C, E, D, H, L & M

Operation → Content of R will add in [A] & final result will store in [A]

[A] ← [A] + [R]

Add. Mode R ≠ M → Regis add.
R = M → indirect add.

Ex: [A] = 25H
[R] = 9CH
ADD R
[A] = C1H

S	Z	AC	P	Cy
1	0	1	0	0

[A] = 11000001 = C1H

(20)

1) ADI 8bit data

IWS → 2byte

Operation:- 8 bit data will get added in accumulator & result will store in ACCumulator.

[A] ← [A] + [8bit data]

Add. Mode immediate add. mode.

2) SUB R

IWS = 1 byte

R → A, B, C, D, E, H, L & M.

Operation:- When inst. will execute content of R will get Subtracted from content of A & result will store in A.

[A] ← [A] - [R]

Add. Mode

Register Add. mode R≠M

Indirect R=M

Ex:-

[A] = 29H

[B] = 35H

SUB B

[A] = 0010 1001

[B] = 0011 0101

[A] = 1111 0100

S	Z	AC	P	Cy
1	0	?	0	1

3) SUI 8bit data IWS = 2byte

8 bit data get subtracted from [A] & result will store in [A]

[A] ← [A] - 8bit data Addressing

• Q: write down one line inst. that make content of ACC-00H regardless of previous value.

(i) SUB A

(ii) MVI A 00H.

(21)

(iv) INR R

IWS = 1 byte

R = A, B, C, D, E, H, L, & M

Operation — when this inst. will execute content of R will increase by 1 & result will store in R

$[R] \longleftarrow [R] + 1_{LSB}$

Ex: B = 21H

INR B

$$[B] = \begin{array}{r} 00100001 \\ +\ 1_{LSB} \\ \hline 00100010 \end{array}$$

[B] = 22H

Add mode — R≠M Reg. add.

R = M Indirect Reg. add. Mode.

(v) DCR R

IWS = 1 byte

R = A, B

Operation: — When this inst will execute content of R will decrease by one & Result will store in R.

$[R] \longleftarrow [R] - 1_{LSB}$

Note: → ADD, ADI, SUI, SUB will affect status of all flag

→ INR & DCR will affect only four flag (S, Z, AC, P)

→ INR & DCR will not affect carry flag

B = FFH

INR B

$$[B] = \begin{array}{r} 11111111 \\ +\ 1 \\ \hline 00000000 \end{array}$$

S Z AC P Cy

0 1 1 1 previous

Group IV - 8-bit Logical operation :-

1) AND operation :-

(22)

(i) ANA R.

IWS = 1-byte

R → A, B, C, D, E, H, L & M.

operation :- when this inst. will execute content of R will get AND operation with [A] bit by bit result will store in A.

Ex:

[A] = 0000 1100 (0CH)

[B] = 5EH

[A] = 10101100 - ACH

[B] = 01011110 - 5EH

[A] = 00001100

(ii) ANI 8-bit data :-

IWS = 2byte

operation :- 8 bit data will get AND operation with content of Accumulator and Result will store in [A]

Add. Mode = Immidiate add.

Ex ANI 00H

(2) OR operation :-

i) ORA R

IWS = 1byte

R = ABCDEHL&M.

operation :- Content of R will get OR operation with contents of A bit by bit & result will store in A.

Ex: MVI A 2.5H

MVI C 1EH

[A] = 2.5H = 00101001

[B] = 1EH = 00011110

[A] → 00111111

if R≠M Register add. Mode
R=M indirect add. Mode.

(iv) ORI 8 bit data
IWS – 2 byte

Operation:- 8 bit data will get OR operation with content of ACC. bit by bit & result will store in [A].

Add. Mode = immidiate

(v) Ex-OR operation →

$Y = A \oplus B = A\bar{B} + \bar{A}B$

(i) XRA R
IWS = 1 byte ; R → A B C D E H L & M.

operation:- When this inst. will execute content of [R] will get EX-OR operation with [A] bit by bit & result will store in [A].

Ex:
[A] = 0101 0011 = 53H
[D] = 1011 0010 = B2H
XRAD → 1110 0001 = [A] = E1H

(ii) XRI 8 bit data :-
IWS = 2 byte

operation:- 8 bit data will get EX-OR operation with content of [A] bit by bit & result will store in [A]

Add. Mode → immidiate Add. Mode.

CMA no operand Mode :- IWS = 1 byte

operation:- When this inst. will execute CMA content of [A] will get complimented bit by bit & result

$[A] \leftarrow \overline{[A]}$

Ex: $[A] = 29H = 0010\ 1001$

CMA

$[A] = 29H = 00101001$

$[A] = D6H = 11010110$

(24)

Ex:- flag can be classified in two categories.

i) Affect according the final result of A & L operation
= S, Z, P.

ii) Affect according the process of A & L operation
= Cy, AC

	S	Z	AC	P	Cy
ANA	✓	✓	1	✓	0
ANI	✓	✓	1	✓	0
ORA	✓	✓	0	✓	0
ORI	✓	✓	0	✓	0
XRA	✓	✓	0	✓	0
XRI	✓	✓	0	✓	0
CMA	X	X	X	X	X

X → not affect

✓ → According the result

0 → Reset

1 → set

→ CMA will not affect any status of any flag.

→ ~~AND, OR, EX-OR, with~~ always reset the ~~Cy~~ Reset carry flag.

→ AND operation always set the AC flag.

→ OR & Ex-OR operation will always reset the AC flag.

Ques

```
MVI A, 2AH
ADD A
ORI AFH
INR A
CMA
HLT
```

after the execution of HLT instruction, status of flag

(25)

Soln

[A] = 2AH

```
[A] =  0 0 1 0  1010
       0 0 1 0  1010
[A] =  0 1 0 1  0100
AFH =  1 0 1 0  1111
       1 1 1 1  1 1 1 1
                1   INR A
       0 0 0 0  0 0 0 0
```

S	Z	AC	P	Cy
0	0	1	0	0
1	0	0	1	0
0	1	1	1	0

↳ status of flag

Ques

```
FF00  MVI A 23H
FF02  MVI B 32H
FF04  XRA B
FF05  ADI 88H
FF07  HLT
```

after execution of above program value of program counter(PC) [B] = ? and PSW (program status word)

```
[A] = 23H = 0010 0011
[B] = 32H = 0011 0010
[A] =       0001 0001
88H =       1000 1000
            1001 1001
```

PC = FF07 + FF08 = FF0F

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

PSW = A1

= 9984 H

Ques:- After the Arithmetic operation status of flag Register BBH Then content of [A] may be

(26)

(a) DBH (b) 75H (c) 65H (d) B6H

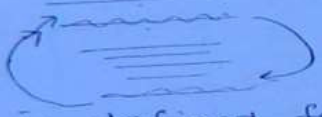
	S	Z	X	AC	X	P	X	CY
	1	0	1	1	1	0	1	1
DB	1	1	0	1	1	0	1	1
75	0	1	1	1	0	1	0	1
65	0	1	1	0	0	1	0	1
B6	1	0	1	1	0	1	1	0

P=0 odd parity
Z=0 & S=1

(d) Auxillary (AC) and carry (CY) are check when process is given only then these parity will find out.

Branch operation :-

- Loop is the special case of branch operation.



- In 8085 3 instruction is defined for branch operation.

1. JUMP
2. CALL
3. RESTART

Jump instruction :-

JMP 16 bit address → Vector Location (unconditional JMP)

IWS = 3 byte

Operation :- When this instruction will execute control will jump at given vector location (address)

Add. Mode → Immediate Add. Mode

Eg:

```
1000  MVI B 27H
1002  MOV D, E
1003  JMP 2918H  ----> 2918
1006  ---------
```

1004 → 18
1005 → 29

(27)

★ Jump instruction is data transfer instruction so status of flag will not affect.

Instruction cycle, Machine cycle and T-state :—

Instruction cycle :-
total time required by execution to execute the instruction is known as instruction cycle.

★ Instruction cycle is the combination of one or more than one machine cycle.

or

Instruction cycle out of 6 machine cycle one or more than one machine macycle will exist.

Machine cycle :—
in 8085 Six type of Machine cycle exist

[1] Memory Read M. cy (R)
[2] Opcode fetch M. cy (F/C)
↳ (Machine code fetch M. Cy)
[3] Memory write M. cy (W)
[4] Input Read M. cy (I)
[5] output write M. cy (O)
[6] Bus idle M. cy (B)

Definition :- total time required by execution to execute different type of task in the execution of instruction.

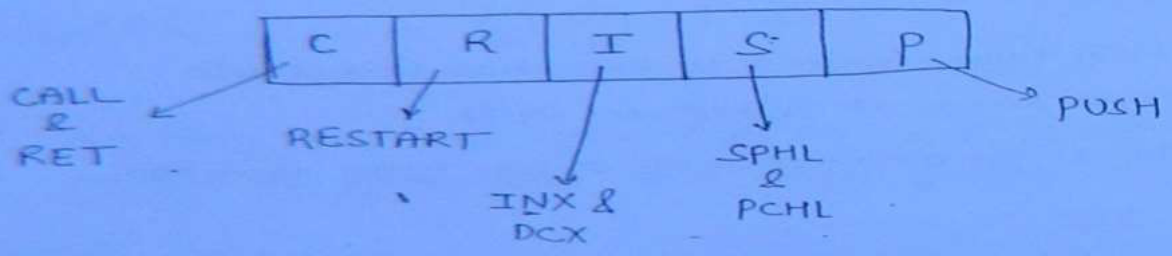
[1]- Fetch Machine cycle (F/S)—

- It is first or only first machine cycle of every inst.
- total time required by execution to fetch opcode (M. cy) from memory
- opcode fetch M. cy is the special case of Memory Read operation.

F = 4T → (238 code)

S = 6T → (8 code)

opcode/machine code required 6T state M. cy



[2] Memory Read Machine cycle (R) :—

- Time required to read 8-bit data from Memory.

$$1R = 3T$$

$$T \text{ state} = \tau = \frac{1}{f}$$

[3]- Memory Write Machine cycle (W) :-

- time required by execution to store 8 bit data in Memory

$$1W = 3T$$

[4]- Input Read Machine cycle (I) :-

- Time required by execution to read 8-bit information from I/P port

$$1I = 3T$$

[5] - o/p Write M.cyle (O) :-

Time required by execution to make available 8 bit data at o/p.

1 O = 3 T

[6] Bus idle Machine cycle :- (B)

during the ~~exprer~~ execution of some special inst. time for which Bus of i/p will be in idle condition and

1 B = 3T

** **this Machine cycle will exist in the execution of DAD inst.**

Inst.	M. Cy.	T-state
MOV B C	F	4T
MVI D, 29H	FR	7T
MOV B M	FR	7T
IN 25H	FRI	10T
OUT 19H	FRO	10T
ADD B	F	4T
ADD M	FR	7T
NOP	F	4T
HLT	F	4T/5T
SUB M	FR	7T
INR D	F	4T
INR M	FRW	10T
ANA M	FR	7T
CMA	F	4T
JUMP if hi add	FRR	10T

(30)

	IO/$\overline{M}$	$\overline{RD}$	$\overline{WR}$	S_1	S_0
Fetch M·Cy	0	0	1	1	1
Memory Read M·Cy	0	0	1	1	0
Memory Write M·Cy	0	1	0	0	1
I/P Read M·Cy	1	1	0	1	0
O/P Write M·Cy	1	1	0	0	1

S_1	S_0	
1	1	→ fetch M·Cy.
1	0	→ Read operation
0	1	→ Write operation.

Conditional JUMP inst. :-

JC	16 bit add.	if Cy = 1
JNC	"	Cy = 0
JZ	"	Z = 1
JNZ	"	Z = 0
JP	"	S = 0
JM	"	S = 1
JPE	"	P = 1
JPO	"	P = 0

IWS ≡ 3 byte

operation:- If condition satisfy then jump instruction will execute otherwise skip it.

→ Add. Mode → Immidiate Add. Mode same as unconditional Jump.

Conditional JUMP → satisfy ≡ FRR ≡ 1
→ Not satisfy ≡ FR = 7T

(31)

→ it is also data transfer inst. so state of flag is not affect

16 bit data Tx. instruction:-

(i) LXI Rp 16 bit data :-

Rp = BC ← DE , HL , SP.
⇓ ⇓ ⇓
B D H

IWS → 3 byte.

- 16 bit data given in the instruction will store in Register pair.

[Rp] ← [16 bit data]

Add. Mode → immidiate Add. Mode.

M. Cycle FRR
LXI B 2011 H

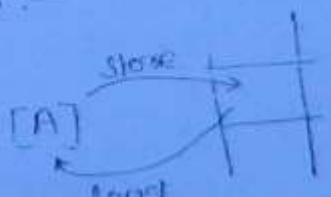
note LXI SP 16 bit data
[S] = 16 bit data.

this inst is use for initilisation state in machine Memory.

LXI SP 11FFH
[SP] = 11FF

(ii) LDA 16 bit address :-

IWS = 3byte



- data available at the Memory add. that is given in inst.

[A] ← [[16 bit add.]]

Add Mode → Direct add. Mode

M.Cy FRRR
LDA 20FC

iii) STA 16 bit add :-

IWS = 3 byte.

Operation: Content of [A] will store at the memory location that is given in instruction.

[A] ⟶ [[16 bit add]].

Ex: STA 29DE H

2000 - STA
2001 - DE
2002 - 29

Machine cycle → FRRW

Add. Mode → direct addressing mode.

iv) LDAX Rp :-

IWS = 1 byte

Operation :- data available at the 16 bit memory location of Rp will load in [A].

Ex: LXI B 2919H
LDAX B
[A] = FC

add. mode →

LDAX Rp → indirect Reg. add. mode (source data).
→ Register add. mode (destination data).

M cycle → FR

v) STAX Rp :-

IWS = 1 byte

Content of [A] will store at Memory location that is content of Register pair.

[A] ⟶ [[Rp]]

STAX Rp → indirect add. (destination data)
→ Register add. (Source data)

(33)

Note →

[1]- for LDAX and STAX possible value of Rp.

Rp = BC
DE

[2]- All above inst. is data transfer instruction so status of Carry flag will not affect.

Ques

```
    MVI   A   F0H
    ORA   A
Loop: INR   A
    JNC   Loop,
    HLT
```

How many time loop will execute.

[A] = F0 = 1111 0000

	Cy
1111 0000	0
[A] → 1111 0000	0
1	
1111 0001	
1	
1111 0010	
1	
.	
.	
1111 1111	
1	
0000 0000	

@ INR will not affect Carry flag.

Loop will execute ∞ time.

Q:- MVI A FOH FR = 7T
ORA A F = 4T
LOOP: INR A F = 4T
JNZ LOOP FRR/FR = 10T/7T
HLT F = 4T.

(34)

If f = 3MHz then find total time of execution of program.

total T-state = 7T + 4T + 15T (14T) + 1 (4T + 7T) + 4T
= 236T
= $236 \times \frac{1}{3}$ μsec
= $\frac{236}{3}$ μsec

16 bit Arithmatic instruction :-

1) INX Rp
IWS = 1 byte
Rp → BC, DE, HL.
Content of Reg. pair will increase by 1 and Result will store in Reg. pair.
[Rp] ← [Rp] + 1_{LSB}
Add. Mode → Reg. Add mode
M. Cy → S (6T)

2) DCX Rp
IWS = 1 byte
Rp → BC, DE, HL
Content of Rp will decrease by 1 and result will store in Rp
[Rp] ← [Rp] − 1_{LSB}

Note :- INX & DCX will not affect status of flag.

(iii) DAD Rp

IWS = 1 byte.

Rp → BC, DE, HL

Content of Rp will get added with [HL] and Result will store in [HL] pair.

→ Reg. Add. Mode.

xxx → M.Cy = F8B ≡ 10T

Note → DAD inst. is affect only one flag that is carry flag

Ques =

1000H	MVI A A1H	[A] = A1H
1002H	LXI H 1007H	[H] = 10H ; [L] = 07H
1005H	SUB M	[A] = 1010 0001
1006H	OUT 05H	[M] = 0000 0101
1008H	HLT	[A] = 1001 1100

9 C

after the execution of this program display the o/p port

9CH Ans

(iv) HL pair is addr. of M and Content of HL pair is the addr that addr data present is M.

Ques:

```
LXI H 2000H
LDA 2002H
XRA M
MOV E,A
MVI D,20H
LDAX D
OUT 01H
```

Memory Location	Data
2000H	00H
2001H	01H
2002H	02H
2003H	03H

after the execution of this program

[H] = 20H , [L] = 00H

[A] = [02]H

[A] = 0000 0010

[M] = 0000 0000

[A] = 0000 0010 = 02H

[E] = 02H [D] = 20H

8-bit logical Rotational instruction:-

Execution of these instruction based upon content of [A].

after the execution of these instruction content of accumulator will rotate by one bit either left or Right as per instruction.

RLC No operand → content of [A] rotate by one bit left without carry.

RAL No operand → Content of [A] Rotate by 1 bit left with carry.

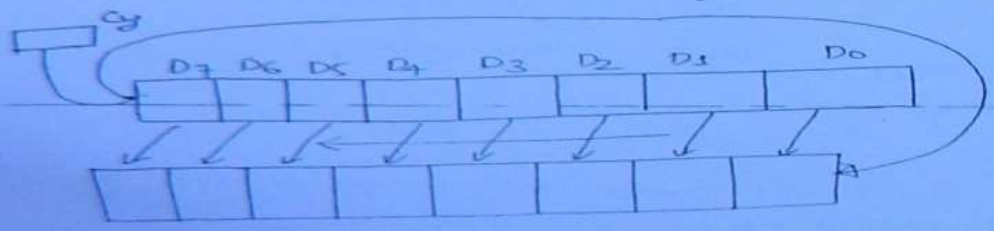
RRC No operand → Content of [A] Rotate by 1 bit Right without carry.

RAR No operand → Content of [A] will Rotate by one bit right with carry.

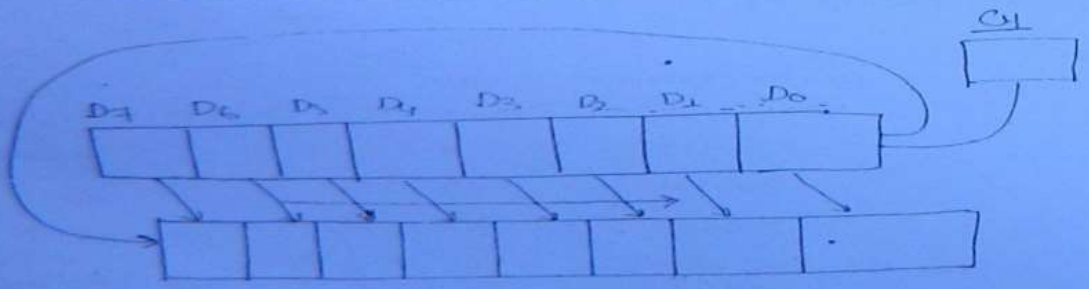
IWS = 1 byte

Operation:- As per inst. Content of [A] will Rotate by one bit.

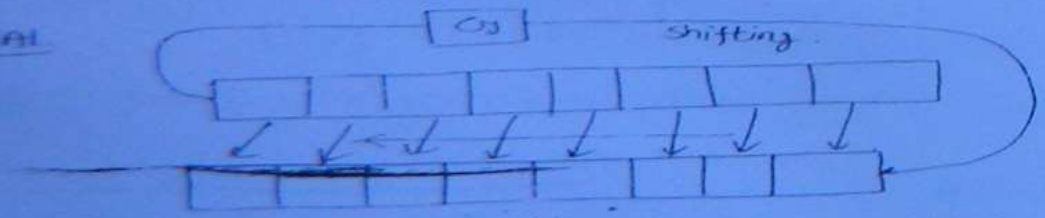
RLC



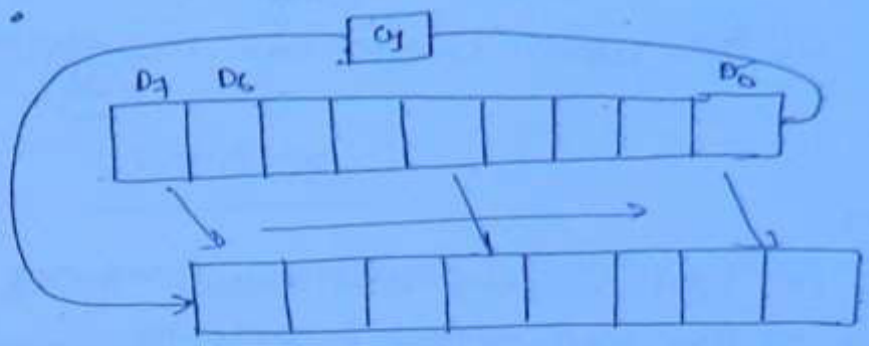
RRC



RAL



RAR



(37)

Note :- Logical rotation will affect only one flag that is carry flag

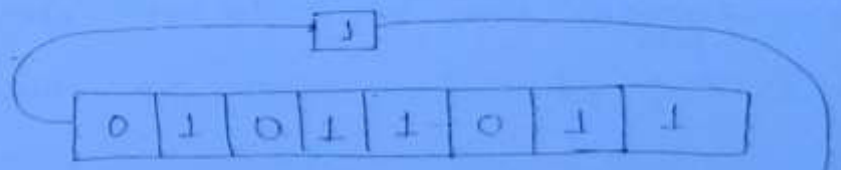
→ Implicit Add. Mode.

→ M. Cy = F

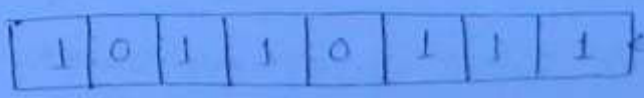
Ques :-
```
MVI A, B9H
ADI A2H
RAL
RLC
HLT
```

What is the content of Acc & status of flag.

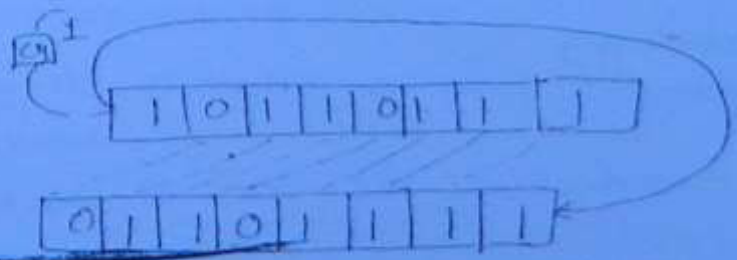
[A] = B9 = 1011 1001
1010 0010
[A] = 0101 1011 Cy = 1



RAL



RLC



7 Logical compair instruction:-

i) CMP R

IWS = 1 byte

R = ABCDEHL & M

operation :- content of [R] will compare with content of [A] and status of flag will affect accordingly.

Note :- compare inst. is just equivalent to sub. inst. of (A-R) and status of flag will affect according result of (A-R) but result (A-R) will discard it means content of [A] and content of [R] will not change.

	Cy	Z
[A] > [R]	0	0
[A] < [R]	1	0
[A] = [R]	0	1

→ Rest of flag affect according the result of (A-R)

R ≠ M Register add. mode

R = M indirect add. mode

M. Cy

R ≠ M = F

R = M = FR

ii) CPI 8 bit data

IWS = 2 byte

8 bit data will compare with content of [A] and status of flag will affect accordingly.

→ immediate add. mode

Note - CMP & CPI will affect status of all flag

STACK

it is group of continuous memory location in main memory. that is used for temporary storage of information during the execution of main program.

- Stack is define by instruction LXI SP 16 bit data.
- add of top of stack always store in stack pointer.
- at a time two byte data can store or retrive from stack.
- if two byte data stored at the top of stack then it grows upwards in numerically decrease order of its address.
- in 8085 two inst are define for store or Retrieve of two byte data at/from top of stack that is PUSH & POP.

PUSH → store

POP → Retrieve

during the execution of CALL subroutine add. of next instruction will store at the top of stack automatically.

PUSH Rp :-

1 byte Rp ⇒ BC, DE, HL, PSW

Operation :- when this inst will execute content of Rp will store at top of stack.

→ after the execution of PUSH inst. content of stack pointer decrease by 2

→ it is data transfer inst. so status of flag will not affect

→ Reg content (x) - there is no conditional push inst.

POP Rp

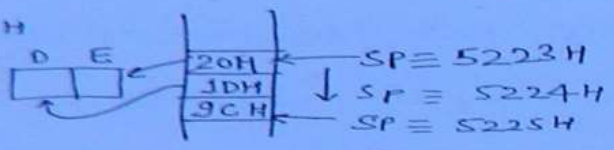
(40)

IWS = 1 byte.

Rp = BC, DE, HL PSW
(B) D H

Operation :- when this inst. will execute two byte data will retrieve from top of stack & store in Rp.

E:
LXI SP 5223H
POP D



D = ?
E = ?

:- After the execution of pop inst. content of SP will ↑ by two.

Add. Mode

POP Rp → indirect add. Mode (source data)
→ Register add. mode (destination data)

M. Cy — FRR.

- it is data transfer inst. so status of flag will not affect.

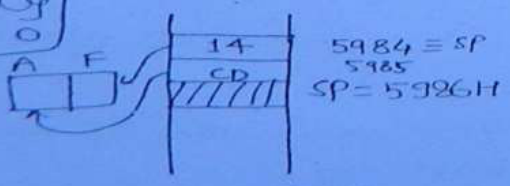
E:- but in POP PSW
status of flag may get affected indirectly.

E: MVI A, 29H
ADD A
LXI SP 5986H
LXI B CDH
14
PUSH B
POP PSW

[A] = 29H = 0010 1001
[A] = 0010 1001
[A] = 0101 0010

S	Z	AC	P	Cy
0	0	1	0	0

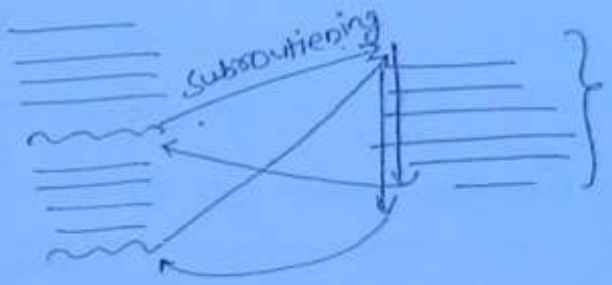
[B] = CDH
[C] = 14H



[A] = CDH
14H = 0001 0100
S Z X AC X P X Cy

Subroutien :- (41)

it is set of inst. that written seperately from main program regarding the task that occur in main program is known as subroutien.



in 8085 two inst. are available for subroutiening.

① CALL inst :-

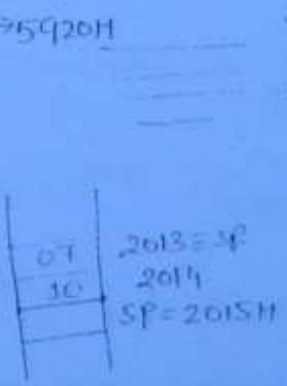
Uncondition CALL inst :-

CALL 16 bit add

IWS = 3byte.

operation :- when this inst will execute control will jump at 16 bit vectors location given in the inst. but before jump add of next inst will store at top of stack.

Ex: 1000 LXI SP 201SH
1003 MOV B C
1004 CALL 5920H
1007 MVI D 18H



PUSH

CALL = PUSH + JMP

Note :- After the execution of call inst content of stack pointer will decrease by two.

*** Note :- immidiate add. mode.

M Cy SRRWW (IRT)

Conditional CALL inst :-

CC	16 bit add.	CALL. CALL inst. will execute if Cy=1
CNC	"	Cy=0
CZ	"	Z=1
CNZ	"	Z=0
CP	"	S=0
CM	"	S=1
CPE	"	P=1
CPO	"	P=0

IWS ≡ 3byte

Operation if condition satisfy then CALL inst. will execute
if condition not satisfy then skip it.

CZ 2011H

CALL 1019H → immediate
→ Register
→ indirect Regi

Add mode — immediate

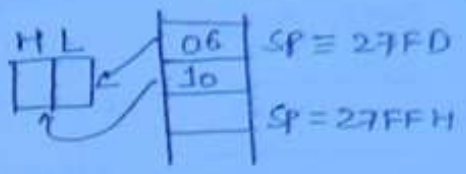
Machine code → Condition satisfy ≡ SRRWW
→ Condition not satisfy ≡ SR

Note ① CALL inst. is data transfer inst so status of flag will not affect
② CALL inst. (conditional in satisfy case (or) unconditional)
is the largest of inst of 8085 & it take 18T

Ques

		SP	HL
1000:	LXI SP 27FFH	(a) 27FF	1006
1003:	CALL 1006H	(b) 27FD	1006
		(c) 27FF	1003
		(d) 27FD	1003

SP = 27FF
HL = 1006



RET inst :-

- unconditional
- conditional

Note → it is last inst. of every subroutien.

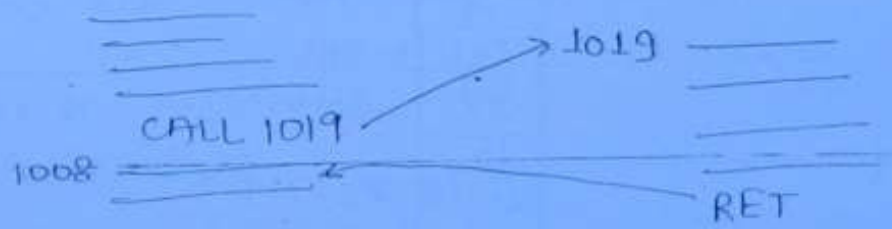
→ RET no operand.

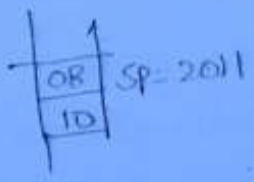
IWS =

operation :- when this inst will execute

(i) two byte data will retrieve from the top of stack & store in program counter.

→ & next inst. fetch from this vector location.





Add. Mode —

- implicit
- indirect

M. cy = 5RR

RET = POP + JMP

Note :- After the execution of RET inst content of SP will ↑ by two

Q:

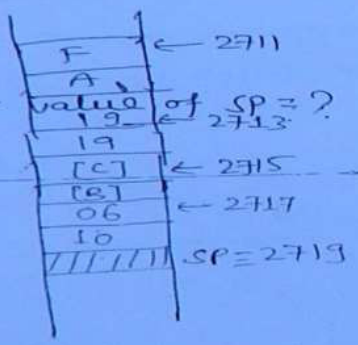
```
1000H: LXI SP 2719H
       CALL 3000H
1003
1006
       ____________
       ____________
       ____________

3000H: LXI H 1919H
       PUSH B
       PUSH H
       PUSH PSW
       LXI SP 3C82H
       POP PSW
       POP H
       POP B
       RET
```

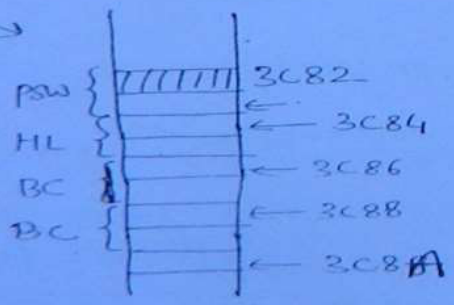
After the execution of program value of SP = ?

SP = 2719

H=19 , L=19

SP = 3C82H





main program ⌐ subroutiens.

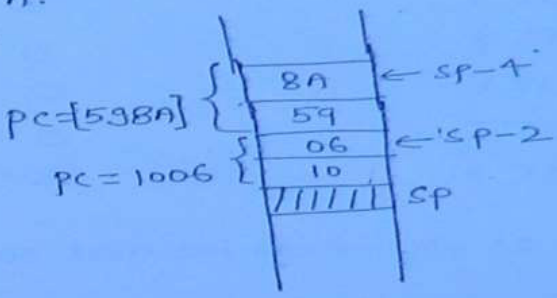
```
1002  MOV B C          5985 : MVI A, 00H
1003  CALL 5985H       5987 : CALL SUB1
1006  MVI  D29         (598A) SUB1 : INR A.
                                    RET
```

[A] = ?

[A] = 00H

[A] = 01

[A] = 02



RESTART

- it is just like as one byte CALL inst.
- it is used when execution stops in interrupts.

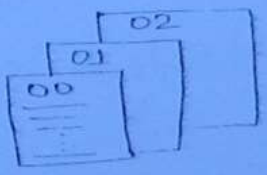
RST N no operand

1 WDS = 1 byte

N = 0, 1, 2, 3, 4, 5, 6, 7.

operation:- when this inst. will execute control will transfer at specific vector location of memory page no. 00H

page NO. line no.
XY WZH



Note:- RESTART is software interrupt.

Int. RSTN	Vector location 00XXH
RST0	0000H
RST1	0008H
RST2	0010H
RST3	0018H
RST4	0020H
RST5	0028H
RST6	0030H
RST7	0038H

(46)

Note :- XX is the hexadecimal conversion of (N×8)

Add. Mode :- Implicit

M. Cy = S

Advance Arithmatic inst :-

ADC R

IWC = 1 byte

R = A, B, C, D, E, H, L & M

Content of R will get added in content of Accumulator with status of Carry flag

$$[A] \leftarrow [A] + [R] + [Cy]_{LSB}$$

Ex. [A] = 21H, [B] = 10H } Cy = 1

ADC B

[A] = 0010 0001

[B] = 0001 0000

[A] = 0011 0001

Add. Mode - R≠M Register add.
R=M indirect.

M. Cy R≠M F
R=M FR.

(ii) ACI 8 bit data.

IWS = 2 byte.

operation:- [A] ← [A] + 8 bit data + [Cy]$_{LSB}$.

→ immediate add. mode.

M. Cy ≡ FR.

(iii) SBB R

IWS = 1 byte

R = ABCDEHL & M.

Content of R will get subtracted from content of Accumulator with status of carry flag.

[A] ← [A] − [R] − [Cy]$_{LSB}$.

R≠M → Register Add. mode

R=M → indirect

Machine Cycle R≠M ≡ F
R=M ≡ FR.

(iv) SBI 8 bit data

IWS = 2 byte.

[A] ← [A] − 8 bit data − [Cy]$_{LSB}$.

M. Cy = FR

Immediate Add. mode.

inst. Related to HL pair :-

(48)

i) LHLD 16 bit add.

IWS = 3 byte.

When this inst. will execute data available at the Memory location that is given in inst. will load in L- and data available at the next memory location will load in H.

[L] ⟵ [16 bit add]

[H] ⟵ [16 bit add + 1]

LHLD 201C H

[H] = 7BH

[L] = 69H.

69H	201CH
7BH	201DH

M-Cy FRRRR

Add-mode direct

1000	LHLD	
1001	1C	
1002	20	

ii) SHLD 16 bit add

IWS = 3 byte

operation :-

[L] ⟶ [16 bit add]

[H] ⟶ [16 bit add + 1]

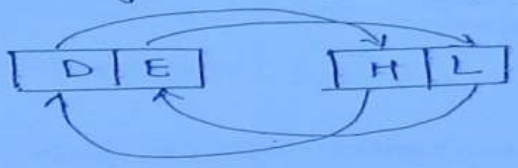
M-Cy (FRRWWW)

1000 SHLD 2019H

(iii) XCHG no operand. (49)

IWS = 1 byte.

When this inst. will execute content of DE region exchange with the content of HL register pair.



Add. Mode → implicit add. ✓
→ Register add.

M.cy - F

(iv) XTHL no operand

IWS = 1 byte.

operation :- when this inst will execute content of HL pair will exchange from top of stack

XTHL = PUSH + POP

M.cy FWWRR

Add. Mode → implicit
→ Register
→ indirect Reg.

(v) SPHL no operand

IWS = 1 byte.

When this inst will execute content of HL pair will copy stack pointer.

[SP] ← [HL]

Note → this inst also used for initialising of stack indirectly.

Add. Mode → implicit ✓
→ Register

M. Cy 5.

(50)

(VI) PCHL no operand.

1 byte.

when this inst. will execute content of HL pair, will copy in PC.

[PC] ← [HL]

Add. Mode → implicit ✓
→ Register.

M. Cy → 5

Note :- All above inst. are data transfer inst. so status of flag with not affe

Que)

```
MVI B 28H     FR   7T
NOP           F    4T
Loop: DCR B   F    4T
JNZ loop      FRR/FR  10T/7T
HLT           F    4T
```

1T = 0.5 μs total time elapsed during the execution of program.

total T = 7T + 4T + 39[14T] + 1[11T] + 4T

= 572T

= 572 × 0.5 μsec

```
MVI B, 38H          FR   7T
Loop2: MVI C, FFH   FR   7T
Loop1: DCR B        F    4T
       JNZ Loop1    FRR/FR  10T/7T
       DCR B        F    4T
       JNZ Loop2    FRR/FR  10T/7T
```

Operating freq. = 2MHz

total time elapsed = ?

(57)

$(38)_H = (56)_{10}$

$(FF)_H = (255)_{10}$

total T-state req. to execute inner loop completely once

$= 255(14T) + 11T = 3567T$

total T for program

$= 7T + 55[7T + 3567T + 4T + 10T]$ (satisfy Loop2) $+ 1[7T + 3567T + 4T + 7T]$ (not satisfy)

$= 200932T$

$T = \frac{1}{2MHz} = 0.5\ \mu sec.$

Que :-

```
LXI H, 2041H    FRR = 10T
MVI A, D6H      FR  = 7T
CMP M           FR  = 7T
MOV B A         F   = 4T
IN 20H          FRI = 10T
SUB B           F   = 4T
OUT 52H         FRO = 10T
INX H           S   = 6T
MOV M, A        FW  = 7T
HLT             F   = 4T
```

(i) How many times cpu executes memory read machine cycle of given program. = 6 times

(ii) How many times cpu perform memory read operation (or) How many times cpu will read data from memory = 11 times

) How many times up perform read operation or how many times $\overline{RD}$ signal will active low during the execution of program. ≡ 17 times.

(N) How many T-state req. of execution of prog ≡ 69T

) f = 3 MHz. So total time req. = ? = 23 μsec.

Some Advance inst :-

i) STC no operand.
IWS = 1 byte.
When this inst. will execute Carry flag will set regardless of previous status. Cy = 1.

→ implicit add. mode.

→ M. Cy – F.

ii) CMC no operand
IWS = 1 byte.
When this inst. will execute status of Carry flag will get complemented

$Cy \leftarrow \overline{Cy}$

Add. mode → implicit

M. Cy → F

Note :- above two inst affect only Carry flag.

iii) DAA

Note :- BCD Numbers
→ Binary coded no. is not binary no.
→ It is binary coded sys decimal

DAA No operand

Ins - 1 byte

(53)

or convert if option not present

this inst will execute then content of Accumulator will adjust in BCD formate by Assuming earlier operation was BCD addition

Add. mode → implicit

M. cy → F

Note :-

① if lower nibble of content of Accumulator is greater than 1001 then 0110 get added in it.

② if lower nibble of [A] is less than or equal to 1001 and Auxillary carry flag set the 0110 will get added in it.

③ if upper nibble of [A] is greater ~~then~~ 1001 then 0110 is added in it

④ if upper nibble of [A] is less than or equal to 1001 & Carry flag is set then 0110 will get added in it.

note → this inst. will affect all flag.

```
6010H: LXI H 8A79H
6013H MOV A, L
6014H ADD H
6015H DAA
60    MOV H A
      PCHL
```

[H] = 8AH [L] = 79H

[A] = 79H. [L] = 79H

```
[A] = 01111001
[H] = 10001010
      ---------
      00000011
[A]   01100110
      ---------
[A] = 01101001 = 69H
```

[H] = 69H

S	Z	AC	P	Cy
0	0	1	1	1
0	0	0	1	0

After the execution of PCHL inst next inst will fetch from

[PC] = 6979H

(a) 6019H

(b) 0379H

(c) 6979H

(d) none of these