8085 Instructions & Programming :-

Instruction → (OPCODE / OPERAND)

A. Instruction of 8085 can be :

a1. One byte instruction (1 Byte)
a2. Two byte instruction (2 Bytes)
a3. Three byte instruction (3 Bytes)

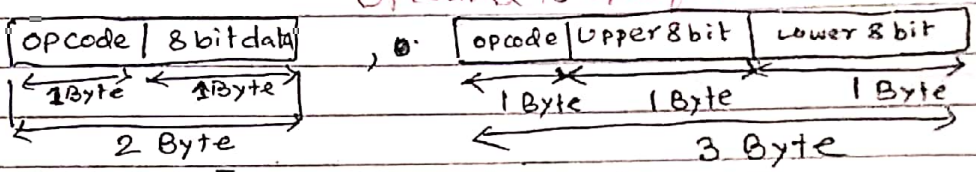
} All instructions come in these groups

Example : MOV C, A | ADD B | CMA | → 1 Byte
MVI A, 35 | → 2 Byte
JMP 2500H | CALL 2500H | → 3 Byte

B. Instruction of 8085 can be : (Based on Addressing Mode (A.M.))

b1 : Immediate Addressing Mode :

Operand is specified within instruction

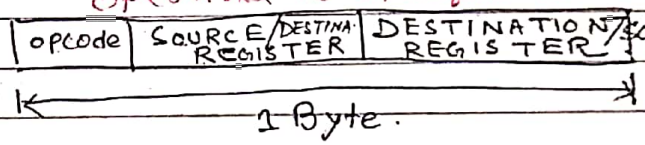


Ex: MVI A, 35H
ADI 06H.

Ex: ~~LDA 2050H~~
LXI H, 2000H.

b2 : Register Addressing Mode :

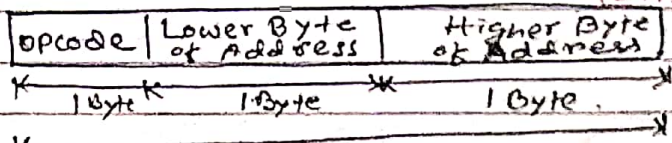
Operand is specified between two registers



Ex: MOV A, B
ADD B.

b3 : Direct Addressing Mode :

Operand is available at Memory location / IO location (Given in the instruction)

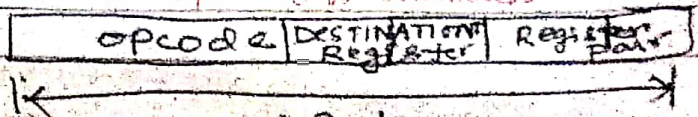


Ex: STA 3000H.
LDA 2500H.

Ex: IN 05H or OUT FFH

b4 : Register Indirect Addressing Mode :

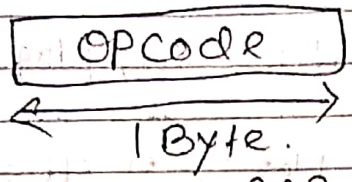
Concern with only Memory specified by register pair where operand is available



Ex: MOV A, M.

b5: Implicit Addressing Mode:

This instruction directly operate on Accumulator and not required any specific operand.



Ex: CMA, RAR, RAL

C. Instruction of 8085 can be: (Based on type of operation).

— ARITHMATIC INSTRUCTIONS —

C1: Addition →

(i) ADD R.

where R is B, C, D, E, H & L

Machine Cycle: OPCODE fetch (OF)

T: 4T

Result: Ac + R ⇒ AC.

Content of Ac & R is added & result store in AC.

Flags- All flag affected.

(ii) ADD M.

where M is 'H-L register pair'.

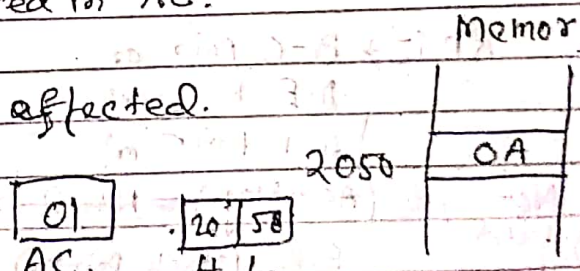
Machine Cycle: O.F. + MEMORY READ

T:- 7T.

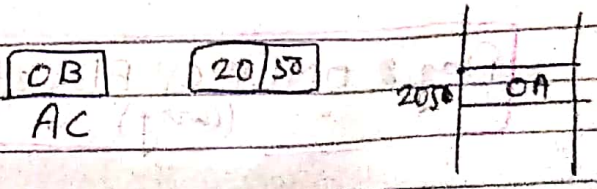
Result: AC + M ⇒ AC

Content of Ac & content of memory location pointed by H-L pair are added & result stored in AC.

Flag - All flags affected.



after execution ADD M,



AC - Accumulator
R - Register

M/c — OPCODE FETCH - OF MEMORY WRITE = MW
MEMORY READ - MR IO Read - IOR, IOW
Write

(iii) ADC R M/c : opcode fetch (OF)

AC + R + CY ⇒ AC. T : 4T.

Content of Ac, Register & Carry flag added. Then result stored into AC.

Flag : All effected.

(iv) ADC M. (Ref. c-ii for M) Concept.

AC + M + CY ⇒ AC. M/c : OF + MR
T : 7T.

Flag : All effected.

(v) ADI 8 bit Data. M/c : OF + MR
T : 7T

AC + Data ⇒ AC.

Content of AC & data is added, then result stored in Ac.

Flag : All effected.

(vi) ACI 8 bit Data M/c : OF + MR
T : 7T

AC + Data + CY ⇒ AC.

Content of Ac, data & carry is added then result stored in Ac.

Flag ⇒ All effected.

(vii) DAD Rp M/c : OF T : 10T **

Add register pair to H and L register then store result into H-L pair.

Rp is ⇒ B-C pair or
D-E pair or
H-L pair or
~~(AC-Flag) = PSW-Pair.~~ (Not used here)
SP (Stack Pointer).

Ex:

20	50
H	L

let's

01	02
B	C

After execution DAD B,

H-L will be | 21 | 52 |

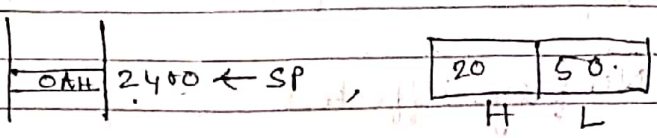
B-C Will be (No change) | 01 | 02 | (B C)

Flag : ONLY CY Flag effected (carry)

Rp — Register pair

** Only for DAD operation DAD takes 10T-states. Now normall O.P. required 4T states. Remaining 6T states are Bus idle state where RD & ALE not activated. Extra T-state is used for internally addition operations (two).

→ DAD SP let,



Then after execution of DAD SP instruction,

HL will be → 44 (H) 50 (L) & SP will be same as before.

C2 Subtration Operation :→

Similar to Addition instructions, subtration instructions also follow the same rules.

(i) SUB R (ii) SUB M (iii) SBB R
(iv) SBB M (v) SUI 8 bit data (vi) SBI 8 bit data

All concept of M/c, T, operation are same except here we do subtraction like

for (i) SUB R ⇒ AC − R ⇒ AC (ii) SUB M ⇒ AC − M ⇒ AC.
(iii) SBB R ⇒ AC − R − CY ⇒ AC (iv) SBB M ⇒ AC − M − CY ⇒ AC
(v) SUI 8 bit data ⇒ AC − 8 bit data ⇒ AC (vi) SBI 8 bit data ⇒ AC − data ⇒ AC

* All flags will be effected.

C3 Increment & Decrement :→

Increment	Decrement
(i) INR R → R is Register.	(i) DCR R. R is Register
(ii) INR M → M is H-L pair.	(ii) DCR M M is H-L pair
Flag : All flags effected except carry	Flag : All flags effected except Carry.
Here Register content or Memory content (Ref. by M). is incremented. & Result is stored at the same place	Here Register content or Memory Cntol (Ref. by m) is decremented & result is stored at the same place.
For INR R → m/c : OF, T : 4T. For INR M → m/c : OF, MR, MW T : 10T	DCR R → m/c : OF T : 4T. DCR m → m/c : OF, MR, T : 10T MW
* If AC ← FF$_H$ then INR A will change AC ← 00$_H$ But CY is not modified.	* If AC ← 00, then DCR A will change or decrese AC by 1. AC ← FF But CY does not change.

64: Increment - Decrement Register Pair

INX Rp where Rp - Reg. Pair

M/c : 0F T : 6T

No flags effected.

Content of Reg. pair will be incremented by 1 and result is stored in the same place

Before: H = 20, L = 11

After execution of INXH: H = 20, L = 12

DCX - Rp where Rp - Reg. Pair

M/c : 0F T : 6T

No flags effected.

Content of Reg. Pair will be decremented by 1 and result is stored in the same place.

Before: H = 20, L = 11

After execution of DCXH: H = 20, L = 10

DAA : Decimal Adjust Accumulator.

It is a special arithmatic instruction of 8085up. The content of Accumulator will change from a binary value to two 4 bits binary Coded decimal digits (BCD).

Only instruction that used 'AC' flag to do Binary to BCD Conversion.

Flags : All flags are effected

operation : let, AC ⇒ 8 bit ⇒ AF_H.

⇒ 1010 1111

Higher 4 bits | lower 4 bits

After DAA execution, AC will change.

- Now, at 1st lower 4 bits are checked.
  if it is > 09 then 06H is added with lower 4 bits
- Next, 4bit bits of higher order are checked
  if it is > 9 or if the CY flag is set then 60H is added with higher 4 bits.

BCD → 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Hexadecimal → 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

6 digit invalid in BCD

- We can add directly 66 to the AC content bcoz both the digit of Ac i.e., 'A', 'F' is greater than 9.

So, A F ⇒ 1010 1111

0110 0110

1 0001 0101

CY

∴ AC ← 15 & CY ← 1. Hence BCD of AF is 1 15

CY AC

(*) This instruction Mostly used in BCD addition.

WAP to add two BCD No. 85 & 68. & store the result to 3000H.

MVI A, 85 BCD — AC ← 85 BCD

ADI A, 68 BCD. — AC ← AC + 68

= 85 + 68

= 1000 0101

0110 1000

1110 1101

E D

So, in 8085 kit AC content is ED.

STA A, 3050 H.

ED

3050

DAA. → 85 BCD | 1110 1101

STA, 3000 H + 68 BCD | +0110 0110

153 BCD | 10101 0011

8 bit

CY (1 bit)

CY AC

1 53

53

3050 H

Here 9 bits is required to store the result.

** Hence CY flag is used to store the 9th bit of BCD addition result.

M/C (i) OF → Opcode Fetch (iv) I/O R → In/out Put Read.
(ii) MR → Memory Read (v) I/O W → In/OUT Put Write
(iii) MW → Memory Write

LOGICAL INSTRUCTIONS.

1. (a) CMP R — Compare 'AC' with data in Register (R)

M/C = 1 (OF) T = 4T

'R' can be - B, C, D, E, H, L, Ac for all logical operations | (AC - R)

• Both the Content will be preserved. The result of comparision is shown by setting the flags as follows -

Operation: if (A) < (reg. R) : Carry flag is set
(A) = (reg. R) : zero flag is set.
(A) > (reg R) : Carry & Zero flag are reset.

Flags: All are effected

(b) CMP M. — • Compare 'AC' with the data stored in a memory location pointed by M (H-L pair)
• Both the Content of AC & memory location will be preserved.

M/C = 2 (OF + MR) T = 7T (4+3)

Operation: Same as before

Flags: All flags effected

2. CPI 8 bit data — Compare immediate (data) with Accumulator

M/C = 2 (OF + MR) T = 7T (4+3)

Operation: as previous.

3. (a) ANA R — Logical ~~and~~ AND register with accumulator & result placed in the Accumulator.

M/C = 1 (OF) T = 4T

Flag: S, Z, P - modified as per result
Cy will reset, AC will set

Example: AC OF , B F0

Now, ANA B

Then, AC - 0000 1111
B - 1111 0000
∴ AC 00
B F0

0000 0000

Flag :-

S	Z	-	Ac	-	P	-	Cy
0	1	0	1	0	0	0	0

⇒ 50_H.

Flag:

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

S - Sign, Z - zero, P - Parity, AC - Auxilliary Carry, CY - Carry.

(b) ANA M - Logical AND ~~register~~ with accumulator & result is placed in the Accumulator. Content of memory location pointed by M (that is always H-L pair)

M/c = 2 (OF + MR)
T = 4+3 = 7.

Flags: S, Z, P Modified as per result. CY will reset & AC will set.

Initial: H L
20 30
AC 01 2050 02
After execution
AC 00
Flags - 01010000

4. ANI 8 bit Data - Logical AND immediate data with accumulator.

M/c = (OF + MR)
2
T = 4+3 = 7

Flags: S, Z, P Modified as per result.
CY = reset & AC = set.

5. (a) ORA R - Logical OR register (R) with Accumulator & placed the result into Accumulator.

M/C = 1 (OF)
T = 4T

Flags: S, Z, P Modified as per result.
CY = reset (=0) & AC = reset (=0)

(b) ORA M - Logical OR the content of memory location pointed by M (which is only H-L pair) with Accumulator & place the result into Accumulator.

M/C = 2 (OF + MR)
T = 7T.

Flags: S, Z, P Modified as Per result.
CY = reset (=0) & AC = reset (=0)

6. ORI 8 bit Data - Logical OR immediate data with Accumulator

M/C = OF + MR
T = 7 (4+3)

Flags: S, Z, P - Modified as per result.
CY & AC will be reset

Example:

AC = OF B = F0

Now, ORAB executed.

0000 1111 - AC
1111 0000 - B
1111 1111 AC = = FF_H.

S = 1, Z = 0, P = 1, AC = 0, CY = 0.

∴ Flag ⇒ 1 0 0 0 0 1 0 0 ⇒ 84H.

Parity - Very important flag. Used to count no. of 1 in a 8 bit no.

LDA 2050 → MOV B, A → ANA B → JPE ~~F~~ (Offset of work) → (set of work) → HLT
True

PSW - Program Status Word.

Ac	Flag

PSW = 16 bits.
Ac + Flag = (8+8) bits.

7. @ XRA R — Exclusive OR register with accumulator & result placed in accumulator.
(R can be → B, C, D, E, H, L & accumulator itself)

M/c = 1 (OP)
T = 4T.

Flag: S, Z, P are modified according the result
CY & AC will be reset.

(b) XRA M — Exclusive OR the content of memory location pointed by H-L pair (= M) with accumulator & result placed into accumulator. Content of the memory does not altered.

M/c = 2 (OP+MR)
T = 7T.

Flag: S, Z, P are modified according to result & CY and AC will be reset.

8. XRI 8 bit data — Exclusive OR immediate with accumulator

M/c = 2 (OP+MR)
T = 4T.

Flag :- S, Z, P are modified according to result. CY & Ac will be reset.

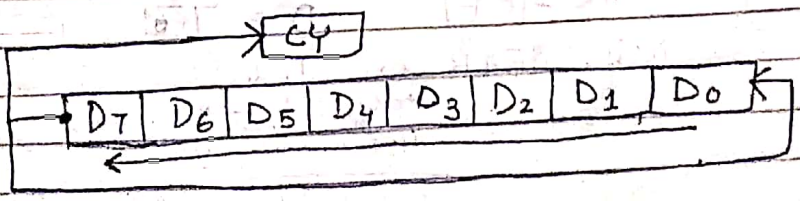
9. CMA : Complement Accumulator.

M/c = 1 (OP)
T = 4T.

Flag : No flag effected.

10. RLC : Rotate Accumulator Left.

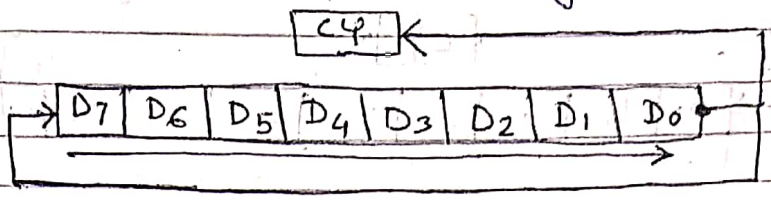
M/c = 1 (OP)
T = 4T.



Flag: ONLY CARRY FLAG MODIFIED & other flags not altered.

11. RRC - Rotate accumulator right.

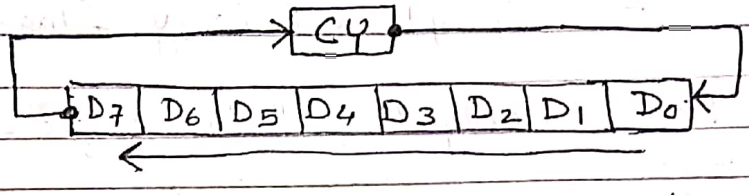
M/C = 1 (OF)
T = 4T.



Flag: only CY flag is modified. others not.

12. RAL - Rotate Accumulator left through carry.

M/C = 1 (OF)
T = 4T.



Flag: Only CY is modified not other flags.

Ex: WAP to shift 16 bit data, 1 bit right. Assume data is in B-C register pair.

Ans.

3A	25
B	C

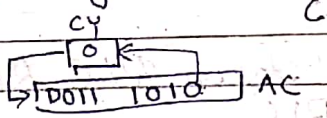
MOV A, B - AC ← 3A

ORA A - Setting CY = 0

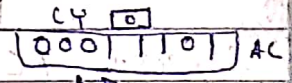
We should make carry reset otherwise wrong CY value will transfer to D15 bit.

RAR - Rotate Ac Right. Put D0 to CY & CY to D7.

that is ⇒



After RAR execute



1D

MOV B, A. → B ← 1D

in CY D8 is saved

(*) We should keep D8 safely at CY and move it to D7 for LSB right shifting.

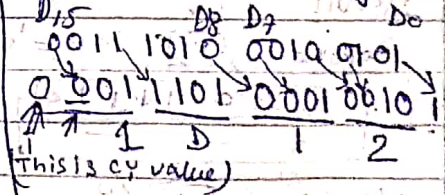
MOV A, C

RAR - Rotate Ac Right

MOV C, A.

1D	12
B	C

Ans



Verification process ⇒ (For why this prog. is important)

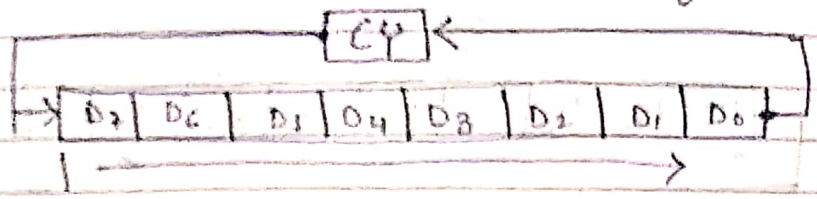
* Step 1 : Take B-C 3A-25.
* Step 2 : Convert it to binary.
* Step 3 : Shift 1 bit right, see result (1D-12)
* Step 4 : Now take only LSB 25 & shift right by RRC. 2 see result ⇒ (92) That is wrong result. Hence verified.

Normal right shift RRC of 25 will give wrong result

0010 0101
1001 0010
9 2

13. RAR - Rotate Accumulator right through carry.

M/C → 1 (OP)
T → 4T



Flag : Only CY flag will be modified.

14. CMC - Complement Carry flag.

M/C → 1 (OP)
T → 4T

Flag : Only CY will be modified & other not changed

15. STC - Set Carry flag.

M/C → 1 (OP)
T → 4T

Flag : Only Carry (CY) flag will chang & no other flag will be changed/effected.

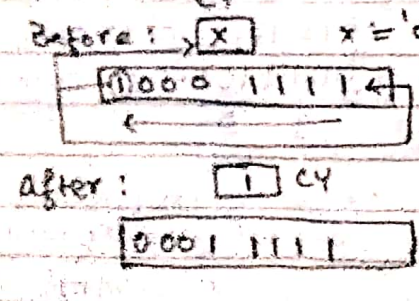
Difference between RLC RAL.

RLC

Suppose, AC is 8FH. & we don't know content of CY flag. Now we execute following instruction:

MVI A, 8FH.
RLC.

Result →

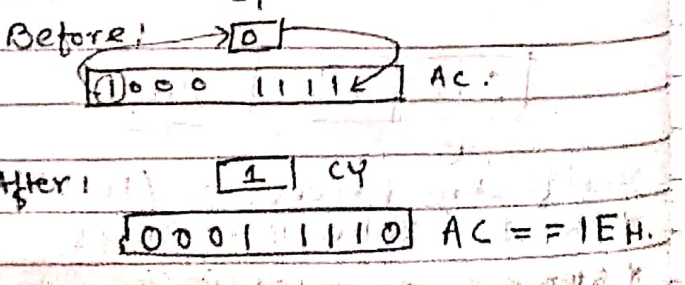


RAL

Suppose, AC is 8FH. ~~& we set CY=0~~ then execute following instruction.

MVI A, 8FH
ORA A → Setting CY = 0.
RAL.

Result →



Diff: Between —

(b)

ORA A.	ANA A.
1. OR (logic) operation accumulator content with accumulator content. Result : store in accumulator	1. AND (logic) operation accumulator with accumulator. Result : store in accumulator.
2. Flag : S, Z, P - change according result CY = reset & AC will be reset also	2. Flag : S, Z, P - change according to result. Here CY and AC will be CY reset & AC set.
3. Register addressing mode.	3. Register addressing mod.
4. M/C = 0.F T = 4T	4. M/C = 0F T = 4T

(c)

CMA	CMC
1. This is complement the accumulator. (Ac) Result: store in Ac.	1. Complement the CY flag only.
2. No flags are effected.	2. Only CY flag is effected.
3. M/C - 0.F T ⇒ 4T Implicit Add. mode	3. M/C - 0F, T = 4T Implicit Add. mode

(d)

CMP B	CMP M.
1. Compare Ac content with Content of register 'B.'	1: Compare Ac with memory Content pointed by M(H-L pair)
2. M/C - 0F. T = 4T	2. Mk = 0F+MR T = 7T
3. Register Add. mode.	3. Register Indirect add. mode

– Branching Instruction –

Branching instructions are →

1. Branching to another location within same program
JUMP

2. Branching to another Program (Sub-routine)
CALL

3. Branching from subroutine program to main Program - RET.

■ JUMP, CALL, RET may be Conditionally or unconditionally. Conditional branching instruction uses the sign, zero, parity and carry flag bits to put up the conditions.

1. JMP 16 bit Address : Jump Unconditionally.

M/C : OF+MR+MR (3)

T : 10T.

Flag : No flag effected.

2. Jump Conditionally :

(a)	JC 16 bit Address.	- Jump on Carry	CY=1
(b)	JNC 16 bit Address.	- Jump on No Carry	CY=0
(c)	JP 16 bit Address	- Jump on Positive	S=0
(d)	JM 16 bit Address	- Jump on Minus	S=1
(e)	JZ 16 bit Address	- Jump on Zero	Z=1
(f)	JNZ 16 bit Address	- Jump on No Zero	Z=0.
(g)	JPE 16 bit Address	- Jump on Parity even	P=1
(h)	JPO 16 bit Address	- Jump on Parity Odd	P=0

All a-h instruction are required 3 byte memory space. If Condition is true then required M/C ⇒ 3 (OF+MR+MR) T ⇒ 10T
Condition is false then " M/C ⇒ 2 (OF+MR) T ⇒ 7T.

3. CALL 16 bit Address : Unconditional.

M/C = 5 (OP, MR, MR, MW, MW)

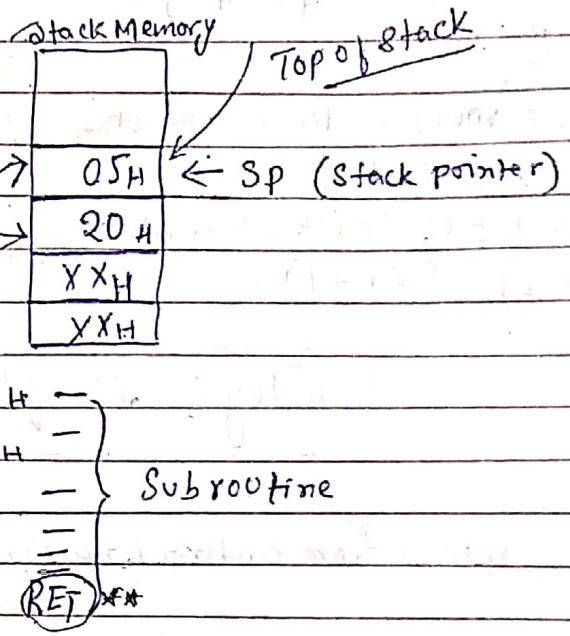
T = 18 (6+3+3+3+3)

Flags : No flag effected.

The program sequence is transferred to the memory location specified by the 16 bit Address given in the operand. Before the transfer, the address of the next instruction after CALL (the content of the Program counter 'PC') is pushed onto the stack. Hence it requires '2' extra Machine cycle i.e. Memory Write (MW) & Memory Write (MW).

Example:

```
2000    .  Instruction 1
2001    .  Instruction 2
2002    CALL 2500
2005    .  instruction 4
2006    .  instruction 5
```



① ② Here two extra Memory Write M/c are to Write 05_H and 20_H into the stack memory.

4. Call Conditionally :

The following conditions can be checked -

CC	16 bit Address	- Call on Carry (CY = 1
CNC	16 bit Address	- Call on No Carry CY = 0
CP	16 bit Address	- Call on Positive S = 0
CM	16 bit Address	- Call on Negative S = 1
CZ	16 bit Address	- Call on Zero Z = 1
CNZ	16 bit Address	- Call on No Zero Z = 0
CPE	16 bit Address	- Call on Parity Even P = 1
CPO	16 bit Address	- Call on Parity Odd P = 0

M/C : 5 (OP, MR, MR, MW, MW) if Condition True - T : 18T

2 (OP, MR) if " false T : 9T

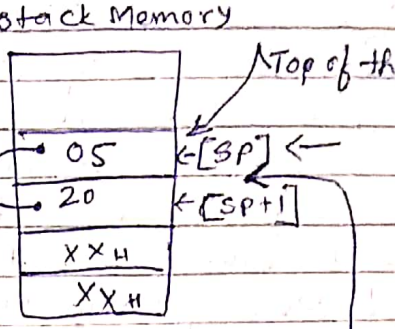
	CALL	RET	JMP
T state → Unconditional	18T (6+3+3+3+3)	10T (6+3+3)	10T (4+3+3)
T state → Conditional	T ⇒ 18. F ⇒ 9 (6+3)	T ⇒ 12T (6+3+3) F ⇒ 9T (6+3)	T ⇒ 10T F ⇒ 7T (4+3)

5. **RET :-** Return from subroutine unconditionally.

The two bytes from the top of the stack are copied into the program counter and program execution begin from that address.

Example:

2000 • instruction1
2001 • instruction2
2002 • CALL 2500H
2005 • instruction 4. ← (2)



2500
2501
2502
2503
2504
2505
2506 RET
Subroutine
(1)

M/C : OF, MR MR.
(3) (4+3+3)

T : 10T

* TWO memory Read operation is required for reading the top of the stack content at SP & (SP+1).

Flag : No flag effected

6. Return from Subroutine Conditionally :

The following conditions are checked → (w.r.to. flag CY, S, Z P).

(a) RC - Return on carry CY=1
(b) RNC - Return on No carry CY=0
(c) RP - Return on Positive S=0
(d) RM - Return on Minus S=1
(e) RPE - Return on Parity Even P=1
(f) RPO - Return on Parity Odd P=0
(g) RZ - Return on zero Z=1.
(h) RNZ - Return on Non-Zero Z=0

M/C →
if condition satisfied (true)
OF + MR + MR
T : 6 + 3 + 3 ⇒ 12 T
If Condition false
M/C - OF + MR
T - 6 + 3 = 9T

Exception : →	CALL 8500	RET	JMP 8500
(Unconditional)	M/C → OF MR MR MW MW	M/C → OF MR MR	M/C → OF MR MR
(Conditional)	M/C → if (true) → 5 M/C (false) → 2 M/C	M/C → if (true) → 3 M/C If false → 2 M/C	M/C → if (true) → 3 M/C. if false → 2 M/C.

7. PCHL - It also comes under branching instruction group. Program Counter (PC) is loaded with content of H-L pair. The PC is 16 bit register & H-L pair also 16 bit. Thus content of H is copied two higher order byte & content of L is copied to lower order byte.

M/C → 1 of 6T ↳ 4 ↳ 2

V. Imp • This instruction is equivalent to unconditional Jump instruction.

• Once the content of HL is copied to PC the Program sequence will transfer to the new address.

Example:

Before: HL → 20|50

2000 • instruction 1
2001 • instruction 2
2002 • LXI H, 2050H. → Loading H-L pair with 2050H address.
2003 • instruction 3
2004 • PCHL → Jumping to 2050 address.
(2005 – 2049) all instructions between these address limits will be ignored
2049 • instruction ##
2050 • instruction **

Jump to 2050H address.

Flags: No flags are effected

* Are the following questions??
How

8. RST 0 – 7 : Restart.

V. Imp It is equivallent to one byte CALL instruction. It transfers program to one of the eight memory location depending upon the number.

This instruction is used in conjuction with interrupts and inserted using external Hardware

Besides, these can be used as software instruction in a program to transfer prog. execution to one of the eight locations.

Instruction restart address for RST.

RST 0	0000H	8 byte
RST 1	0008H	8 byte
RST 2	0010H	8 byte
RST 3	0018H	8 byte
RST 4	0020H	8 byte
RST 5	0028H	8 byte
RST 6	0030H	8 byte
RST 7	0038H	8 byte

M/C = 3 (OF, MR, MR)
T = (6+3+3) ⇒ 12T.

Flag : No. flag effected

* Some special Instructions –

(a) XCHG → Exchange H-L with D-E

M/C = 1
T = 4T
Flag = No flag effected.

- The content of register H is exchanged with the content of register D.
- The content of register L is exchanged with the content of register E.

Before: H L = 20 50, D E = 30 50 — after execution of XCHG → H L = 30 50, D E = 20 50

(b) XTHL → Exchange H-L with top of stack.

M/C = 5
OF, MR, MR, MW, MW
T ⇒ 16T

Flag : No flag effected.

Example: Before: H L = A2 50, SP = 2095

Stack Memory

2095	38
2096	67

After execution of XTHL, H L = 67 38, SP = 2095

Stack memory:

2095	50
2096	A2

M/C for XTHL

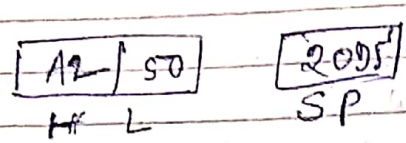
- OF – is essential for all instruction. (To decode meaning of instruction)
- MR – Reading content of 2095 (that is : 38H)
- MR – Reading the content of 2096 (that is : 67)
- MW – Writing the Content of H into 2096 (that is A2)
- MW – Writing the Content of L into 2095 (that is 50)

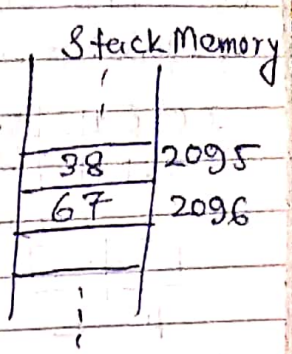
(c) SPHL → Copy H and L registers to Stack Pointer.

M/C = 1
T = 6T

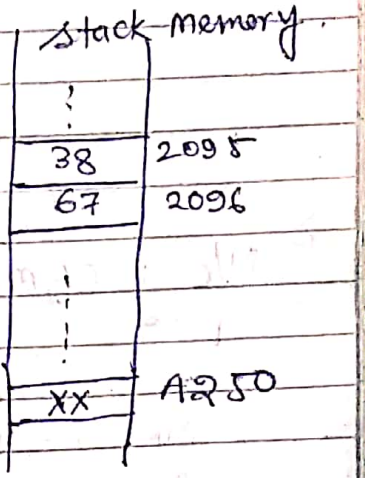
Flag: No Flag effected.

Example: Before





After execution of SPHL instruction,



Imp
So difference is the only SP register value is changed here with respect to instruction XTHL where the contents are exchanged.
Also Note: H-L content still unchanged before & After the execution of SPHL instruction.

(d) LDA 16 bit Address:

OF+MR+MR
M/C ⇒ +(MR) ← To read the content of memory address provided in the instruction by 16 bit add.
T ⇒ 13T

Before: AC [ ] 2050 [0A] LDA 2050
After: AC [0A] 2050 [0A]

Flag — Not effected

(e) STA 16 bit Address:

M/C ⇒ OF+MR+MR
+(MW) ← To write the Ac content into the memory address provided by 16 bit Address.
T ⇒ 13T

Before: AC [0B] 2051 [ ] STA 2051
After: AC [0B] 2051 [0B]

Flag: Not effected

(f) LDAX — B/D reg. pair

NOTE: H-L pair not used

M/C: OF, (MR) ← Read the memory content pointed by either B-C or D-E pair. & put into Ac.

T: 7T Flag: Not effected

(g) STAX B/D register pair

Note: H-L pair Not used.

M/C: OF+(MW) ← Write the memory pointed by either B-C or D-E pair by the value of Accumulator

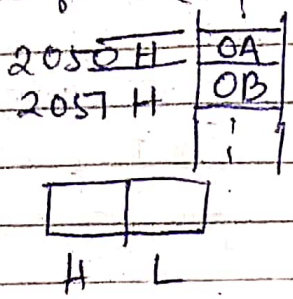
T: 7T Flag: Not effected

(8) LHLD 16 bit Address

(i) Load H & L registeres direct

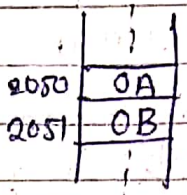
(ii) LHLD 2050

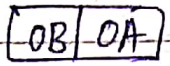
Before : (Memory)



After :

Memory



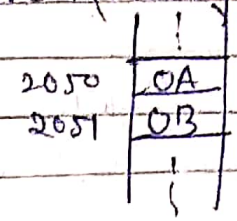


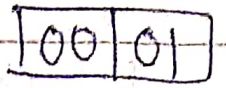
(9) SHLD 16 bit Address

(i) Store H & L registeres direct

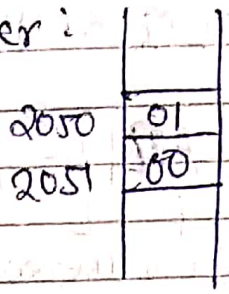
(ii) SHLD 2050

Before : (Memory)

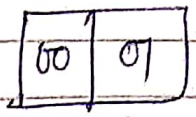




After :



Execution of SHLD 2050.



(8) M/C : OF, MR, MR, MR, MR.

T : 16T

OF : Decode meaning of ins.

MR : Read lower add byte 50H

MR : Read upper add. byte 20H

MR : Read memory content of 2050

MR : Read memory content of 2051

(9) M/C : OF, MR, MR, MW, MW

T : 16T

OF : Decode meaning of instruction

MR : Read lower add. byte 50H

MR : Read upper add. byte 20H

MW : Write 'L' content to 2050 H

MW : Write 'H' content to 2051 H

(7) LXI Rp, 16 bit data

M/C = OF, MR, MR,

T => 10T

Flag : No flag effected.

Rp → B-C ✓
D-E ✓
H-L ✓
SP ✓

X PSW X X

* { LXI H, 2500 / PCHL. } ← PC ⇐ HL.

Program Counter will be loaded with H-L pair value. This is same instruction as "JMP 2500".

JMP 2500

Prog. will directly jump to 2500H memory location.