4 INSTRUCTION SET

4.1 THEORY OF INSTRUCTIONS

Crucials

1. Organization of instruction
2. Function of mnemonics and operands
3. Useful in programs

An instruction is a combination of mnemonics and operand. It can be a command which is used to perform specific task or operation on the operands. In some cases the operand of the instruction is not available, so in this case instruction depends either upon the meaning of the mnemonics or the contents of accumulator.

Example **:** MOV A,B

In this example left part of the instruction is called **mnemonics** and right side that is A,B is called the **operand** on which operation is to be performed (data transfer).

Example **:** HLT

The command stops the execution of the program. It can be seen that the meaning of the mnemonics itself is the function of the instruction. No operand is available in this case.

Instructions can be classified into basic five groups. A flowchart is provided which gives the details of the groups :

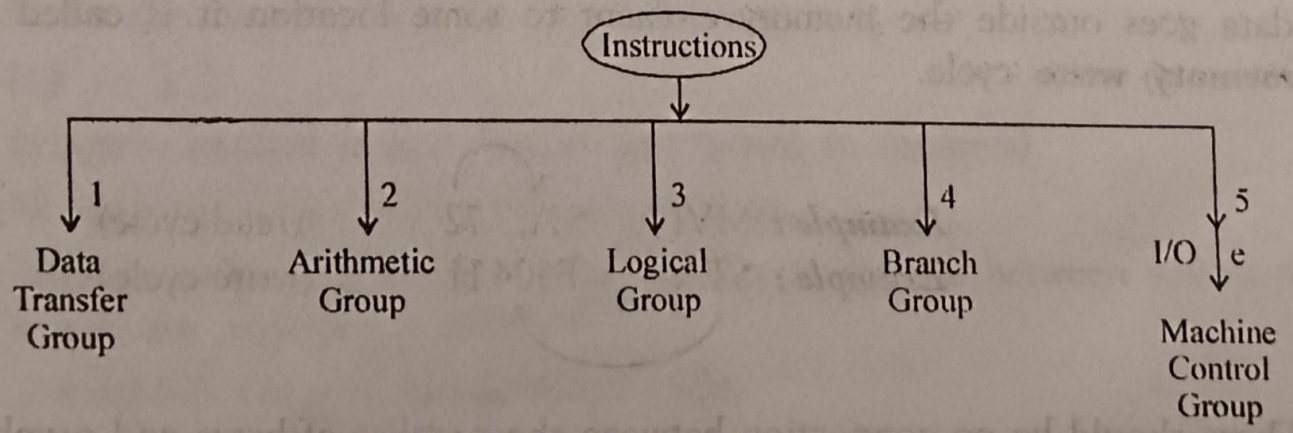


Fig. 4.1 : Instruction Classification

Six parameters are required for the proper execution of a particular instruction :

1. Byte
2. Flags
3. Group
4. Addressing Modes
5. Machine Cycles
6. T–states

Clues

(1) Change the mnemonics and obtain the multiple instructions of the same group e.g. MOV A,B; MVI A, 72; ADD B; ADI 72; ADC B; ACI data etc.

(2) Symbolic representation includes [ ] as the contents of register and [[ ]] as the contents of memory location whose address is in the register pair which is denoted by r_p.

(3) I → Immediate
L → Load
PC → Program Counter (16–bit)
ST → Store
M → 16–bit address in H–L pair
X → Register Pair (16–bit)
R → Register (8–bit) A, B, C, D, E, H, and L
SP → Stack Pointer (16–bit)

(4) Always first–byte is the opcode (8–bit) of the particular instruction. (MOV A,B) second–byte is for the 8–bit data or second and third–byte is for the lower order address and higher order address inside the operand (a) MVI A, 72 (b) STA F504 H.

(5) Define the first machine cycle always as an opcode fetch cycle.
Define the second and third machine cycle of the instruction only when 8–bit data/address or 16 bit data/address is visible inside the operand of the instruction.

(6) Up cycles can be defined as :
(a) Opcode fetch cycle → 4 T–states/5 T–states/6 T–states
(b) Read cycle → 3 T–states
(c) Write cycle → 3 T–states

When data goes inside the register it is called **read cycle** or **memory read cycle**. When data goes outside the memory element to some location it is called **write cycle** or **memory write** cycle.

Example : MVI A, 72 (read cycle)
Example : STA F504 H (write cycle)

(7) There should be no connection between the number of bytes and number of machine cycles of the particular instruction.

(8) See the balance of the weight of the bits 8/16 between the source and destination to decide the machine cycles and addressing modes of the particular instruction.

(9) (a) Presence of I in the mnemonics indicates immediate addressing mode.
Example : MVI A, 72.
(b) Presence of M always indicates for indirect addressing mode.
Example : ADD M.
(c) Balanced source and destination either 8–bit/16–bit indicates register addressing mode.
Example : MOV A,B
DAD H.

(10) (a) No flags are affected in case of data transfer and branch group instructions.
(b) All flags are affected in case of arithmetic instructions except INX and DCX and except carry in INR and DCR.
(c) All flags are affected in case of logical instructions but in case of
AND → CS → 0 state
AC → 1 state
Learn (AND & AC starts from A so AC → 1)
OR → CS → 0 state
AC → 0 state
Learn (OR states from 0 so CS & AC → 0)
XOR → CS → 0 state
AC → 0 state
Learn (conditions for OR and XOR are same)

(11) Up carry is always shown by 1 (set) and no carry is shown by 0 (reset)

4.2 CLASSIFICATION OF INSTRUCTION GROUP

4.2.1 Data Transfer Group

(1) MOV r_1, r_2 (MOV A,B)

Move the content of source r_2 to destination r_1.

(a) $[r_1] \leftarrow [r_2]$.
(b) 1 byte – opcode (since data is not known in registers).
(c) No flags as data transfer reflects no flags.
(d) Register addressing mode – since balance is defined between source and destination, which are registers of 8–bit.
(e) 1 machine cycle – opcode fetch cycle.
(f) 4 T–states – execution of the instruction is 4 clock periods.

(2) MOV r, M

Move the content of 16–bit memory to 8–bit register.

(a) [r] ← [[M]] or [r] ← [[HL]].

(b) 1 byte – opcode (since data is not known in register pair HL).

(c) No flags are affected as data transfer reflects no flags.

(d) Indirect Addressing Mode (since M is present).

(e) 2 machine cycles – (i) opcode fetch cycles – 4 T.
(ii) memory read cycle – 3 T.

(f) 7 T–states – 4 T–states for opcode fetch.
– 3 T–states for reading and move contents to register (destination).

(3) MVI r, Data

Move immediately 8–bit data to destination register r.

(a) [r] ← data.

(b) 2 bytes – 1st byte is the opcode.
– 2nd byte is the 8–bit data present inside the operand. (data is known).

(c) No flags are affected as data transfer reflects no flags.

(d) Immediate Addressing Mode (since I is present)

Immediate data present is moved inside register r (destination).

(e) 2 machine cycles – opcode fetch cycle – 4 T.
– memory read cycle – 3 T.

(f) 7 T–states – 4 T–states for opcode fetch.
– 3 T–states for reading the data inside accumulate/ register r.

(4) MVI M, Data

Move immediately the data to memory.

(a) [[H–L]] ← data.

(b) 2 bytes – 1st byte is the opcode.
– 2nd byte is the 8–bit data present inside the operand (data is known).

(c) No flags are affected as data transfer reflects no flags.

(d) Only instruction in data transfer group having indirect as well as immediate addressing mode (I is present and M is present).

(e) 3 machine cycles – opcode fetch – 4 T–state.
– memory read (bytes) – 3 T–states.
– memory write (data or byte) – 3 T–states.

(f) 10 T–states.

(5) LXI rp, 16–bit Data

Load register pair immediately with data 16–bit present in the operand. Initially the 16–bit is an address in hexadecimal, but since immediately is used so after loading it becomes data (higher order and lower order).

(a) $[r_p]$ ← data 16–bit, $[r_h]$ ← 8 MSB's $[r_L]$ ← 8 LSB's.

(b) 3 bytes – 1st byte is the opcode
– 2nd byte is the lower order 8 LSB's
– 3rd byte is the higher order 8 MSB's

(c) No flags are affected as data transfer reflects no flags

(d) Immediate Addressing Mode (since I is present)

(e) 3 machine cycles – opcode fetch cycle – 4 T–states
– memory read (LOB) – 3 T–states
– memory read (HOB) – 3 T–states

(f) 10 T–states – 4 T–states opcode fetch
– 3 T–states (LOB) *i.e.* $[r_L]$ – 8 LSB's
– 3 T–states (HOB) *i.e.* $[r_h]$ – 3 MSB's

(6) LDA Address

Load accumulator with the contents of 16–bit memory address

(a) [A] ← [addr]

(b) 3 bytes – 1st byte is the opcode
– 2nd byte is the lower order address
– 3rd byte is the higher order address

(c) No flags are affected as data transfer effects no flags

(d) Direct Addressing Mode (address directly learn defined)

(e) 4 machine cycles – opcode fetch – 4 T–states
– memory read – 3 T–states
– memory read – 3 T–states
– memory read – 3 T–states

(f) 13 T–states – 4 T–states – opcode fetch
– 3 T–states for (LOB) 8–bit
– 3 T–states for (HOB) 8–bit
– 3 T–states for reading register contents

(7) STA Address

Store the contents of accumulator at 16–bit address.

(a) [addr] ← [A]

(b) 3 bytes – 1st byte is the opcode
– 2nd byte is the lower order address
– 3rd byte is the higher order address

(c) No flags are affected as data transfer effects no flags.

(d) Direct Addressing Mode (address directly defined)

(e) 4 machine cycles
- opcode fetch – 4 T–states
- memory read – 3 T–states
- memory read – 3 T–states
- memory write – 3 T–states

(f) 13 T–states
- 4 T–states – opcode fetch
- 3 T–states (LOB) 8–bit
- 3 T–states (HOB) 8–bit
- 3 T–states for writing register contents

(8) LHLD Address

Load register pair HL directly with the contents of two consecutive memory locations (address) and (address+1)

(a) [L] ← [addr], [H] ← [addr+1]

(b) 3 bytes – 1st byte is the opcode
- 2nd byte is the lower order address
- 3rd byte is the higher order address

(c) No flags are affected as data transfer effects no flags

(d) Direct Addressing Mode (address directly defined)

(e) 5 machine cycles
- opcode fetch – 4 T–states
- memory read – 3 T–states
- memory read – 3 T–states
- memory read – 3 T–states
- memory read – 3 T–states

(f) 16 T–states
- 4 T–states – opcode fetch
- 3 T–states –(LOB) 8–bit
- 3 T–states – (HOB) 8–bit
- 3 T–states – read data at address in L
- 3 T–states – read data at (address + 1) in H

(9) SHLD Address

(Store register pair HL contents at two come active locations at address and (address+1)

(a) [addr] ← [L], [addr + 1] ← H

(b) 3 bytes – 1st byte is the opcode
- 2nd byte is the lower order address
- 3rd byte is the higher order address

(c) No flags are affected.

(d) Direct Addressing Mode(address directly defined)

(e) 5 machine cycles
- opcode fetch – 4 T–states
- memory read – 3 T–states
- memory read – 3 T–states
- memory write – 3 T–states
- memory write – 3 T–states

(f) 16 T–states
- 4 T–states opcode fetch
- 3 T–states (LOB) 8–bit
- 3 T–states (HOB) 8–bit
- 3 T–states (write data from L to address)
- 3 T–states (write data from H to address+1)

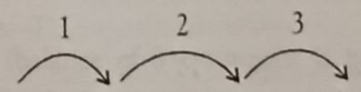
(10) LDAX r_p

Load the contents of memory location whose address is specified in the register pair in accumulator. It is used only for BC and DE pair.

(a) [A] ← [[r_p]]

(b) 1 byte – opcode is the byte

(c) No flags are affected.



(d) Indirect Addressing Mode – r_p → address → content → A; (indirect process) 3 steps.

(e) 2 machine cycles
- opcode fetch – 4 T–states
- Read cycle – 3 T–states

(f) 7 T–states
- 4 T–states opcode fetch
- 3 T–states to read data from r_p in accumulator.

(11) STAX r_p

Store the content of accumulator in memory location whose address is specified in the register pair. It is used only for BC and DE pair

(a) $[[r_p]] \leftarrow [A]$

(b) 1 byte – opcode is the byte.

(c) No flags are affected.

(d) Indirect Addressing Mode – A → content → address → r_p; (indirect process) 3 steps

(e) 2 machine cycles
- opcode fetch – 4 T–states
- write cycle – 3 T–states

(f) 7 T–states
- 4 T–states opcode fetch
- 3 T–states to write data from A to r_p

(12) XCHG

Exchange the contents of HL pair with DE pair always. Initial address after loading in HL and DE pair becomes data for HL and DE pair so one bracket is used.

(a) [HL] ↔ [DE]

(b) 1 byte – opcode

(c) No flags are affected

(d) Register Addressing Mode (balance of 16 bits is maintained between HL and DE)

(e) 1 machine cycle – opcode fetch – 4 T–states

(f) 4 T–states – opcode fetch.

4.2.2 Arithmetic Group

(1) ADD r

Add the contents of register to hidden accumulator and store the reset in accumulator (in built register)

(a) [A] ← [A] + [r]

(b) 1 byte – opcode

(c) All flags are affected

(d) Register Addressing Mode (balance of weight is maintained between A and r)

(e) 1 machine cycle – opcode fetch – 4 T–states

(f) 4 T–states – opcode fetch

(2) ADD M

Add the content of memory location addressed by H–L pair to the content of accumulator and store the result in accumulator.

(a) [A] ← [A] + [[HL]]

(b) 1 byte – opcode

(c) All flags are effected

(d) Indirect Addressing Mode (M is present)

(e) 2 machine cycles
- opcode fetch – 4 T–states
- memory read – 3 T–states

(f) 7 T–states
- 4 T–states opcode fetch
- 3 T–states to read the data inside accumulator.

(3) ADI data

Add immediately the data to the contents of accumulator and store the result in accumulator

(a) [A] ← [A] + data

(b) 2 bytes – 1st byte is the opcode
- 2nd byte is the data inside the instruction.

(c) All flags are affected.

(d) Immediate addressing mode (since I is present)

(e) 2 machine cycles
- 1st opcode fetch – 4 T–states
- 2nd memory read – 3 T–states.

(f) 7 T–states
- 4 T–states for opcode fetch
- 3 T–states to read the data in accumulator

(4) ADC r

Add the contents of register along with carry to the contents of accumulator and store the result in accumulator.

(a) [A] ← [A] + [CS] + [r]

(b) 1 byte – opcode

(c) Register Addressing Mode (balance of 8–bit weight is maintained between A and r)

(d) All flags are affected

(e) 1 machine cycle – opcode fetch – 4 T–states

(f) 4 T–states – opcode fetch cycle

(5) ADC M

Add the contents of memory location addressed by H–L pair with carry to the contents of accumulator and store the result in accumulator.

(a) [A] ← [A] + [[H–L]] + [CS]

(b) 1 byte – opcode

(c) Indirect Addressing Mode – (M is present)

(d) All flags are affected

(e) 2 machine cycles – opcode fetch cycle – 4 T–states
– memory read – 3 T–states.

(f) 7 T–states – 4 T–states – opcode fetch
– 3 T–states – read the content (data) of accumulator.

(6) ACI data

Add with carry immediate data to the content of accumulator.

(a) [A] ← [A] + [data] + [CS]

(b) 2 byte – 1st byte is the opcode
– 2nd byte is the data inside the instruction

(c) All flags are affected.

(d) Immediate Addressing Mode (since I is present)

(e) 2 machine cycles – opcode fetch– 4 T–states
– memory read – 3 T–states

(f) 7 T–states – 4 T–states opcode fetch
– 3 T–states – read the contents or data of accumulator.

(7) DAD r_p

Direct addition of contents of register pair to the contents of H–L pair and store the result in H–L pair. Always contents are added to H–L pair. For 8–bit addition ADD is used and DAD is used for 16–bit addition.

(a) [HL] ← [HL] + [r_p]

(b) 1 byte – opcode

(c) Only carry flag is affected (result is becoming 16–bit).

(d) Register Addressing Mode (balance of weight of source and destination is maintained which is of 16 bit)

(e) 3 machine cycles

- opcode fetch – 4 T–states
- In 2nd machine cycle lower byte of register pair is added to the L (lower register) of H–L and stored in ALU from which it is transferred to L [$RP_L \rightarrow A$]
  [L → TMP] (temporary register) [A + TMP → ALU → L] 3 T–states
- In 3rd machine cycle higher byte of register pair is added to the H (higher register) of H–L and stored in ALU from which it goes to H [$RP_h \rightarrow A$]
  [H → TMP] [A+ TMP → ALU → H]

(f) 10 T–states – opcode fetch – 4 T–states
– defined above (3 T) + (3 T) = 6 T

(8) DAA

Decimal adjust the contents of accumulator.

(a) 1 byte – opcode

(b) All flags are affected

(c) 1 machine cycle – opcode fetch – 4 T–states

(d) 4 T–states – opcode fetch condition

- It can be used after ACI, ADC, ADD and ADI.
- Since the result goes in accumulator so 8–bit data is only adjusted.
- The result in A is adjusted and final result is obtained in decimal system.
- When the sum result comes in between A and F in hexadecimal 6 is added to LSB's and MSB's, which is called adjustment.
- When auxiliary carry is generated from bit D_3 to D_4 during addition adjustment of + 6 is made to LSB
- When carry is generated from D_7 to D_8 adjustment of +6 is made to the MSB.

All the feature of subtraction and addition are same so a comparative study of subtract instructions is given below :

Table 4.1 : Comparative Study of Subtract Instructions

	Instruction	Symbol	Flags	Byte	A.M	M/C	T–state
1	SUB r	[A]←[A]–[r]	All	1	Register A.M	Opcode Fetch	4 T
2	SUB M	[A]←[A]–[[H–L]]	All	1	Indirect A.M	(1) Opcode (2) Memory Read	4 T 3T
3	SBB r	[A]←[A]–[r]–[CS]	All	1	Register A.M	(1) Opcode Fetch	4 T
4	SBB M	[A]←[A]–[[H–L]]–[CS]	All	1	Indirect A.M	(1) Opcode (2) Memory Read	4 T 3 T
5	SUI Data	[A]←[A]–data	All	2	Immediate A.M	(1) Opcode (2) Memory Read	4 T 3 T
6	SBI data	[A]←[A]–[data]–[CS]	All	2	Immediate A.M	(1) Opcode (2) Memory Read	4 T 3 T

(9) INR r

Increment the contents of register by 1.

(a) [r] ← [r] + 1

(b) 1 byte – opcode

(c) All flags are affected except carry flag.

(d) Register Addressing Mode.

(e) 1 machine cycle – opcode fetch – 4 T–states

(10) INR M

Increment the content of memory location addressed by H–L pair by one.

(a) [[HL]] ← [[HL]] + 1.

(b) 1 byte – opcode.

(c) All flags are affected except carry.

(d) Indirect Addressing Mode (since M is present).

(e) 3 machine cycles – opcode fetch – 4 T–states
– memory read – 3 T–states
– memory write – 3 T–states

(f) 10 T–states – opcode fetch – 4 T–states
– read the data of incremented value in ALU – 3 T–states
– write the data from ALU to [HL] pair – 3 T–states

(11) DCR r

Decrement the content of register by one.

(a) [r] ← [r] – 1.

(b) 1 byte – opcode fetch.

(c) All flags are affected except carry flag.

(d) Register Addressing Mode.

(e) 1 machine cycle – opcode fetch cycle – 4 T–states

(12) DCR M

Decrement the content of memory location addressed by H–L pair by 1.

(a) [[HL]] ← [[HL]]–1

(b) 1 byte – opcode fetch.

(c) All flags are affected except carry flags.

(d) Indirect Addressing Mode (since M is present).

(e) 3 machine cycles – opcode fetch – 4 T–states
– memory read – 3 T–states
– memory write – 3 T–states

(f) 10 T–states – opcode fetch – 4 T–states
– read the data of decremented value in ALU – 3 T–states
– write the data from ALU to [HL] pair – 3 T–states

(13) INX r_p

Increment the content of register pair r_p by one. Initially this content is address but after initializing in r_p it becomes data or content.

(a) $[r_p] \leftarrow [r_p]+1$ (only one bracket is used).

(b) 1 byte – opcode fetch.

(c) Only instruction in which no flags are affected.

(d) Register Addressing Mode.

(e) 1 machine cycle – 6 T–states

(f) 6 T–states – T_1 – ALE is high
– T_2, T_3 read the opcode
– T_4, $T_5 \rightarrow [r_p + 1] \rightarrow [r_p]$

(14) DCX r_p

Decrement the content of register pair r_p by one. Initially this content is address but after initializing in r_p it becomes data or content.

(a) $[r_p] \leftarrow [r_p] - 1$ (only one bracket is used)

(b) 1 byte – opcode fetch.

(c) Only instruction in which no flags are affected.

(d) Register Addressing Mode.

(e) 1 machine cycle – 6 T–states

(f) 6 T–states – T_1–ALE is high
– T_2, T_3–read the opcode
– T_4, T_5 – $[r_p \rightarrow 1]$–$[r_p]$

4.2.3 Logical Group

(1) ANA r

AND the contents of register with the contents of accumulator and store the result in A

(a) [A] ← [A] ∧ [r]

(b) 1 byte – opcode

(c) All flags are affected. [CS = 0, AC = 1].

(d) Register Addressing Mode (balance of weight of 8 bit is maintained between source and destination).

(e) 1 machine cycle – opcode fetch – 4 T–states

(f) 4 T–states – opcode fetch – 4 T–states

(2) ANA M

AND the contents of memory location addressed by H–L pair with accumulator and store the result in accumulator.

(a) [A] – [A] ∧ [[HL]]
(b) 1 byte – opcode
(c) All flags are affected [CS set to 0, AC set to 1].
(d) Indirect Addressing Mode (since I is present).
(e) 2 machine cycles – opcode fetch cycle – 4 T–states
– memory read – 3 T–states
(f) 7 T–states – 4 T–states opcode fetch
– read the ANDed contents in accumulator–3 T–states

(3) ANI data

AND immediately the data with the contents of accumulator and store the result in accumulator.

(a) [A] ← [A] ∧ data (∧ is the AND operator)
(b) 2 bytes – opcode – 1st byte
– data – 2nd byte
(c) All flags are affected [CS set to 0, AC set to 1].
(d) Immediate Addressing Mode – (since I is present).
(e) 2 machine cycles – opcode fetch – 4 T–states
– memory read – 3 T–states
(f) 7 T–states – 4 T–states for opcode fetch
– 3 T–states to read the ANDed immediate data in ACC.

(4) ORA r

OR the contents of register r with the contents of accumulator and store the result in accumulator.

(a) [A] ← [A] ∨ [r] [OR → 1 + 1 = 1] (∨ is the OR operator)
ADD → 1 + 1 = 0 carry 1]
(b) 1 byte – opcode
(c) All flags are affected [CS → 0, AC → 0]
(d) Register Addressing Mode.
(e) 1 machine cycle – opcode fetch – 4 T–states

(5) ORA M

OR the content of memory location whose address is in H–L pair to the content of accumulator and store the result in accumulator.

(a) [A] ← [A] ∨ [[HL]]
(b) 1 byte – opcode
(c) All flags are affected [CS → 0, AC → 0].
(d) Indirect Addressing Mode (since M is present).
(e) 2 machine cycles – opcode fetch – 4 T–states
– memory read – 3 T–states

(f) 7 T–states – 4 T–states opcode fetch
– read the ORed contents inside the A – 3 T–states

(6) ORI data

OR immediately the data with the contents of accumulator. And store the result, in accumulator.

(a) [A] ← [A] ∨ data [∨ is OR operator]
(b) 2 bytes – 1st byte – opcode
– 2nd byte – data
(c) All flags are affected [CS → 0, AC → 0].
(d) Immediate Addressing Mode (I is present).
(e) 2 machine cycles – opcode fetch – 4 T–states
– memory read – 3 T–states
(f) 7 T–states – 4 T–states opcode fetch.
– 3 T–states to read the ORed data in accumulator.

All the condition for OR and XOR are same so comparative study tabel for XOR is defined below. (∀ is the XOR operator)

Table 4.2 : Comparative Study of OR and XOR

Instruction	Symbol	Flags	Byte	A.M	M/C	T–State
XRA r	[A]←[A] ∀ [r]	All CS→0 AC→0	1 Opcode	Register A.M	Opcode	4 T
XRA M	[A]←[A] ∀[[HL]]	All CS→0 AC→0	1 Opcode	Indirect A.M	(1) Opcode (2) M. Read	4 T 3 T
XRI data	[A]→[A] ∀ data	All CS→0 AC→0	2 byte (1) Opcode (2) Data	Immediate A.M	(1) Opcode (2) M. Read	4 T 3 T

(7) CMA

Complement the contents of accumulator (1 → 0 and 0 → 1)

(a) A ← [$\overline{A}$]
(b) 1 byte – opcode
(c) No flags are affected
(d) Implicit Addressing Mode (working on the contents of accumulator)
(e) 1 machine cycle – opcode fetch cycle – 4 T–states

(8) CMC

Complement the carry status.

(a) CS ← [$\overline{CS}$]

(b) 1 byte – opcode

(c) Only carry flag is affected. No other flags are affected.

(d) 1 machine cycle – opcode fetch – 4 T–states.

(9) STC

Set the carry status flag to 1

(a) [CS] ← 1

(b) 1 byte – opcode

(c) Only carry flag is affected. So other flag is affected.

(d) 1 machine cycle – opcode fetch – 4 T–states

Table 4.3 : Comparative Study of CMP Instruction

Instruction	Symbol	Flags	Byte	A.M	M/C	T–state
CMP r	[A] – [r]	All	1 Opcode	Register A.M	Opcode fetch	4 T
CMP m	[A] – [[HL]]	All	1 Opcode	Indirect A.M	(1) Opcode (2) M. Read	4 T 3 T
CPI data	[A] – data	All	2 (1) Opcode (2) Data	Immediate A.M	(1) Opcode (2) M. Read	4 T 3 T

Table 4.4 : Difference between CMP and SUB

CMP	SUB
(1) Difference is performed but A is not modified.	(1) Difference is performed and A is modified
(2) Logical Group	(2) Arithmetic Group
(3) Original content of A can be obtained	(3) Original content of A is lost.

(10) RLC

Rotate or shift the content of accumulator left by one bit without carry.

(a) [D_n+1] ← [D_n], [D_0] ← [D_7], [CS] ← [D_7]

(b) 1 byte – opcode

(c) Only carry flag is affected.

(d) Implicit Addressing Mode (no operand is there, works on accumulator)

(e) 1 machine cycle – opcode fetch – 4 T–states

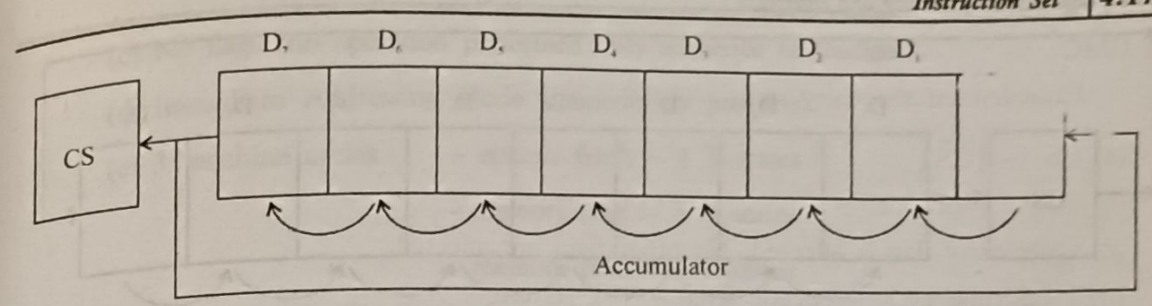


Fig. 4.2

Use : (1) To multiply the numbers by two (MSB → 0)

(2) To check whether bit has carry or no carry.

(11) RRC

Rotate or shift the contents of accumulator right by one bit without carry

(a) [D_7] ← [D_0], [CS] ← [D_0], [D_n] ← [D_n+1]

(b) 1 byte – opcode

(c) Only carry flag is affected.

(d) Implicit Addressing Mode (no operand is present, works on accumulator).

(e) 1 machine cycle–opcode fetch – 4 T–states

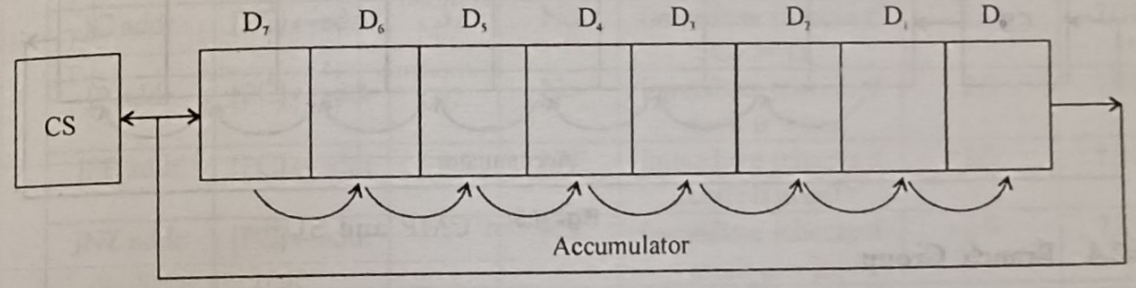


Fig. 4.3

Use : (1) To divide the numbers by two [MSB→0]

(2) To check whether the bit has carry or no carry.

(12) RAL

Rotate the contents of accumulator left by one bit with carry.

(a) [D_n+1] ← [D_n], [CS] ← [D_7], [D_0] ← CS

(b) 1 byte – opcode

(c) Only carry flag is modified.

(d) Implicit Addressing Mode.

(e) 1 machine cycle – opcode fetch – 4 T–states.

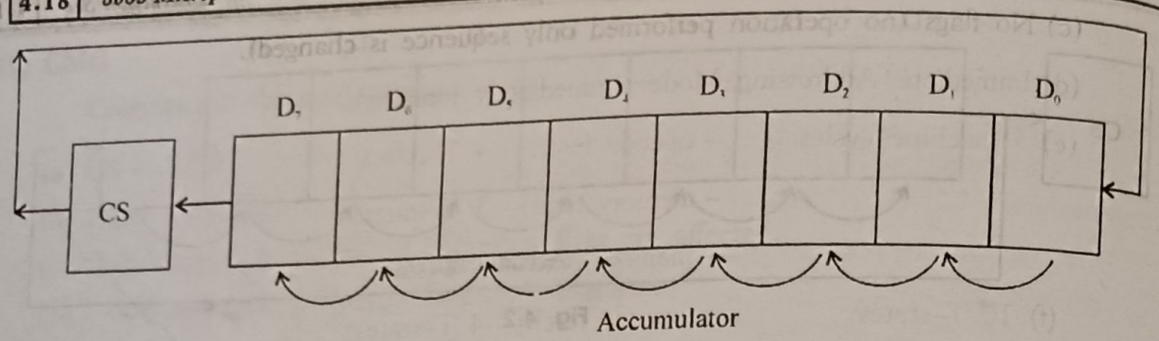


Fig. 4.4

(13) RAR

Rotate or shift the contents of accumulator right by one bit with carry

(a) $[D_n] \leftarrow [D_n + 1]$, $[CS] \leftarrow [D_0]$, $[D_7] \leftarrow [CS]$

(b) 1 byte – Opcode

(c) Only carry flag is affected.

(d) Implicit Addressing Mode

(e) 1 machine cycle – opcode fetch – 4 T–states

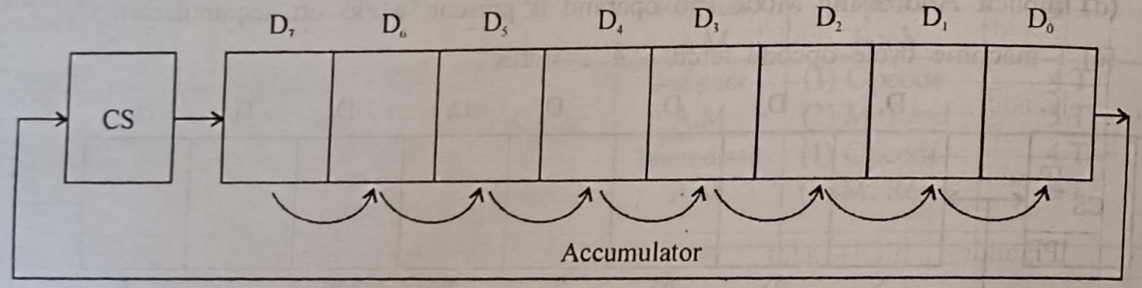


Fig. 4.5

4.2.4 Branch Group

- Instructions of this group change the normal sequence of the program.
- No flags are affected for this group.
- This group is divided into **conditional** and **unconditional** instructions.

(A) Unconditional

(1) JMP Address (16–bit)

Jump unconditionally to the instruction specified by the address in the operand of the instruction.

(a) [PC] ← Address [PC is program counter (16–bit)]

(b) 3 bytes
- 1st byte opcode
- 2nd byte [Lower order address]
- 3rd byte [Higher order address]

(c) No flags (no operation performed only sequence is changed).

(d) Immediate Addressing Mode (immediately jumping).

(e) 3 machine cycles
- opcode fetch – 4 T–states
- memory read – 3 T–states
- memory read – 3 T–states

(f) 10 T–states
- opcode fetch – 4 T–states
- lower order read – 3 T–states
- higher order read – 3 T–states

This instruction is used to jump internally or to form a loop inside the main program.

(B) Conditional Jump Instructions

Table 4.5 : Comparative Study of Jump Instructions

Instruction	Symbol	Byte	Flags	Addressing Mode	M/C	T–state
JZ addr	[PC]←addr	3	No	Immediate (checks if result is 0)	2/3	7/10
JC addr	[PC]←addr	3	No	Immediate (checks if CS = 1)	2/3	7/10
JP addr	[PC]←addr	3	No	Immediate (checks if result is +ve)	2/3	7/10
JPE addr	[PC]←addr	3	No	Immediate (checks if parity is even)	2/3	7/10
JNZ addr	[PC]←addr	3	No	Immediate (checks if result is ≠ 0)	2/3	7/10
JNC addr	[PC]←addr	3	No	Immediate (checks if no carry)	2/3	7/10
JM addr	[PC]←addr	3	No	Immediate (checks if result is –ve)	2/3	7/10
JPO addr	[PC]←addr	3	No	Immediate (checks if parity is odd)	2/3	7/10

- This instruction checks the condition such as zero, carry, even, odd etc. and then jumps.
- If desired condition is true a loop is formed and 3 machine cycles and 10 T–states are required.
- If desired condition is not true control comes out of the loop and 2 machine cycles and 7 T–states are required.

(1) CALL

- Instruction of this group change the normal sequence of the program externally mainly in the subroutines.
- No flags are affected for this group.
- This group is divided into conditional and unconditional instructions.

(2) CALL addr

Call unconditionally to the subroutine outside the main program specified by the address in the operand of the instruction

(a) [[SP – 1] – 1] ← [PCH], [[SP – 2]–2] ← [PCL], [SP] ← [SP]–2 and [PC] ← addr

(b) 3 bytes
- 1st byte (opcode)
- 2nd byte (LOA)
- 3rd byte (HOA)

(c) No flags (no operation performed only sequence is changed).

(d) Immediate Addressing Mode (immediately jumping).

(e) 5 machine cycles
- opcode fetch – 6 T–states
- memory read – 3 T–states
- memory read – 3 T–states
- memory write – 3 T–states
- memory write – 3 T–states

(f) 18 T–states
- 6 T–states opcode fetch.
- Memory read lower order address 3 T–states (8–LSB).
- Read higher order address 3 T–states (8–MSB).
- Memory write contents of stack pointer register [SP – 1] are placed on the address bus. The higher order byte of the (PCH) is placed on the address bus and stored in the location [SP–1]. At this instant stack pointer register becomes [SP–2]. (3 T–states).
- Memory write the contents of stack pointer register [SP–2] on the address bus. The lower order byte of the program counter (PCL) is placed on the data bus and stored at location [SP–2], (3 T–states).

Imp

- Used for call subroutine.
- The address of the instruction after call is placed in the stack.
- The contents of the SP goes on decremented by 2.
- Jumps to the starting address of subroutine specified by second byte and third byte.
- Always associated by RET.

Conditional Call Instructions

Table 4.6 : Comparative Study of Call Instruction

Instruction	Symbol	Byte	Flags	A.M	M/C	T–state
CZ addr	[[sp]–1]← [PCH]	3	No	Immediate	2/5	9/18
CC addr	[[sp]–2]←[PCL]	3	No	Immediate	2/5	9/18
CP addr	[PC]←addr	3	No	Immediate	2/5	9/18
CPE addr	[SP]←[SP] –2	3	No	Immediate	2/5	9/18
CNZ addr	Same for all Instructions	3	No	Immediate	2/5	9/18
CNC addr		3	No	Immediate	2/5	9/18
CM addr		3	No	Immediate	2/5	9/18
CPO addr		3	No	Immediate	2/5	9/18

- These instructions check the conditions such as zero, carry, even parity, odd parity and then calls the subroutine.
- If the desired condition is true, execution takes 5 machine cycles and 18 T–states. If condition is not satisfied. 2 machine cycles and 9 T–states are required for the execution.

(3) RET

This instruction returns from subroutine to the main program unconditionally.

(a) [PCL] ← [[SP]], [PCH] ← [[SP]+1], [SP] ← [SP]+2

(b) 1 byte – Opcode

(c) No flags

(d) Register Indirect A.M

(e) 3 machine cycles
- opcode fetch – 4 T–states
- memory read – 3 T–states
- memory read – 3 T–states

(f) 10 T–states
- 4 T–state opcode fetch.
- The contents of the stack pointer register are placed on the address bus. From the stack top data is fetched and stack pointer register is upgraded to next location (3 T–states).
- Next data byte is copied which is higher order byte from the stack and SP is modified to next location. (3 T–states).

Imp

- RET saves back the address from the stack pointer to the program counter.
- Increment the SP register by two to form a new stack top.
- Return unconditionally from subroutine.

(4) RST n

These instructions are 1 byte call instruction which transfers the control from the program execution to a specific location on page 00H. Content of the program counter is saved in the SP.

(a) [[SP] – 1] ← [PCH], [[SP] – 2] ← [PCL], [SP] ← [SP] – 2, [PC] ← 8 × n.

(b) 1 byte – opcode

(c) No flags

(d) Indirect A.M

(e) 3 machine cycles – opcode fetch 6 T–states
– memory read (PCH) – 3 T–states
– memory read (PCL) – 3 T–states

The address is defined by multiplying the n by 8.

Table 4.7

	Instruction	Restart Location	
1.	RST0	(1) 0000	0000*8 = 0000
2.	RST1	(2) 0008	0001*8= 0008
3.	RST2	(3) 0010	0002*8=00016=0010
4.	RST3	(4) 0018	0003*8=0024=0018
5	RST4	(5) 0020	0004*8=0032=0020
6.	RST5	(6) 0028	0005*8=0040=0028
7.	RST6	(7) 0030	0006*8=0048=0030
8.	RST7	(8) 0038	0007*8=0056=0056

(5) PCHL

This Instruction jumps to the memory location specified by H–L pair.

(a) [PC] ← [H–L], [PCH] ← [H], [PCL] ← [L]

(b) 1 byte.

(c) No flags.

(d) Register Addressing Mode (balance is maintained between (PC) and (HL) which are of 16–bits).

(e) 1 machine cycle – opcode fetch – 6 T–states

– The contents of register H is moved to higher order 8–bit of register PC. The contents of register L is moved to the lower order 8–bit of register PC.

4.2.5 Stack I/O and M/C Group

(1) IN Port Address

This instruction moves the data at the port (8 bits) which is outside 8085 to the accumulator.

(a) [A] – [Port]

(b) 2 bytes – opcode – 1st byte
– port address – 2nd byte

(c) No flags – (type of data transfer) no operation is performed.

(d) Direct Addressing Mode (address directly specified).

(e) 3 machine cycles – opcode fetch – 4 T–states
– memory Read – 3 T–states
– I/O read – 3 T–states

(f) 10 T–states – 4 T–states opcode fetch
– read for the byte no 2nd (3 T–states)
– read I/O for the data at the port (8–bit in accumulator)

Imp

– 2nd Memory cycle is memory oriented so IO/$\overline{M}$ defined by $\overline{M}$.

– 3rd Memory cycle is I/O oriented so IO/$\overline{M}$ defined by IO.

(2) OUT Port Address

This instruction moves the data from the accumulator to the 8–bit port outside microprocessor specified in the instruction.

(a) [port] ← [A]

(b) 2 byte – opcode – 1st byte
– port address – 2nd byte

(c) No flags

(d) Direct Addressing Mode (address directly specified)

(e) 3 machine cycles – opcode fetch – 4 T–states
– I/O write – 3 T–states

(f) 10 T–states – 4 T–states opcode fetch
– memory read for byte no 2nd (3 T–states)
– I/O write for the data from accumulator to port 8 bit (3 T–states).

Imp

– 2nd Memory cycle is memory oriented so IO/$\overline{M}$ defined by $\overline{M}$

– 3rd Memory cycle is I/O oriented so IO/$\overline{M}$ defined by IO

(3) Push r_p

This instruction pushes or moves the content of register pair defined in the instruction to the stack. Stack pointer register decrements and higher order register contents copied to

[SP – 1]. Again [SP] is decremented to [SP–2] and lower order register contents are copied to [SP – 2].

(a) [[SP] – 1] ← [r_h], [[SP]– 2] ← [r_l], [SP] ← [SP] – 2

(b) 1 byte – opcode fetch

(c) No flags

(d) Register Indirect

(e) 3 machine cycles
- opcode fetch – 6 T–states
- memory write – higher byte – 3 T–states
- memory write – lower byte – 3 T–states

(f) 12 T–states
- 6 T–states for opcode fetch
- write the higher byte into stack (3 T–states) (r_h)
- write the lower byte into stack (3 T–states) (r_l)

(4) Push PSW

This instruction pushes the processor status word into stack. Accumulator contents are moved to [SP–1] after decrementing by one. Again SP is decremented and contents of flag register are moved to [SP–2].

(a) [[SP] – 1] ← [A], [[SP] – 2] ← flag register, [SP] ← [SP] – 2.

(b) 1 byte – opcode.

(c) No flags.

(d) Indirect Addressing mode.

(e) 3 machine cycles
- opcode fetch cycle – 6 T–states
- memory write – 3 T–states
- memory write – 3 T–states

(f) 12 T–states
- 6 T–states opcode fetch
- write accumulator contents into [SP–1] 3 T–states
- write lower order flag contents into [SP–2] – 3T–states

(5) POP PSW

This instruction retrieves the contents from the stack to PSW.

(a) PSW ← [[SP]]
[A] ← [[SP]+1]
[SP] ← [[SP]+2]

(b) 1 byte – opcode

(c) No flags

(d) Indirect Addressing Mode

(e) 3 machine cycles
- opcode fetch – 4 T–states
- memory read – 3 T–states
- memory read – 3 T–states

(f) 10 T–states
- 4 T–states opcode fetch
- reading the contents in accumulator from [SP+1] requires 3 T–states
- reading the contents in flag register from [SP] requires 3 T–states

Imp

- Contents of the two top locations are copied into register pair.
- Stack pointer is incremented by two (step by step).

(6) POP rp

This instruction retrieves the contents from the stack to the specified register pair

(a) [r_l] ← [[SP]], [r_h] ← [[SP]+1], [SP] ← [[SP]+2]

(b) 1 byte – opcode

(c) No flags

(d) Indirect Addressing Mode

(e) 3 machine cycles
- opcode fetch – 4 T–states
- memory read – 3 T–states
- memory read – 3 T–states

(7) HLT

Stops the execution of program.

(a) 1 byte – opcode

(b) No flags

(c) Implicit Addressing Mode.

(d) 1 machine cycle – opcode fetch – 5 T–states.

(8) XTHL

This instruction exchanges the contents of register L with the data of top of the stack. Also the contents of the register H are unchanged. With the data below the stack top.

(a) [L] ↔ [SP], [H] ↔ [SP+1]

(b) 1 byte – opcode

(c) No flags

(d) Register Indirect Addressing Mode

(e) 5 machine cycles
- opcode fetch – 4 T–states
- memory read – 3 T–states

– memory read – 3 T–states

– memory read – 3 T–states

– memory read – 3 T–states

(9) SPHL

This instruction moves the contents of H–L pair to the stack pointer.

(a) [HL] ← [SP]

(b) 1 byte

(c) No flags

(d) Register Addressing Mode (balance of weights is maintained between (HL and SP) which is of 16–bit)

(e) 1 machine cycle – opcode fetch – 6 T–states

(10) NOP

This instruction performs no operation and can be used to create a delay of 4 T–states or its multiples.

(a) 1 byte – opcode

(b) No flags

(c) Implicit Addressing Mode

(d) 1 machine cycle – opcode fetch – 4 T–states.

4.3 PROGRAM STRUCTURE

A program is written, which consists of some instructions having mnemonic and operands and same logic. This logic is the basic components on which the type of program is made which is called the structure of the program. Program structure can be used to write very long programs. If the same criteria have to be implemented for micro computers, short programs can also be made. All the logic structures used in program structure have signal entry and single accist which can be nested form.

A program can be made using three types of basic structures :

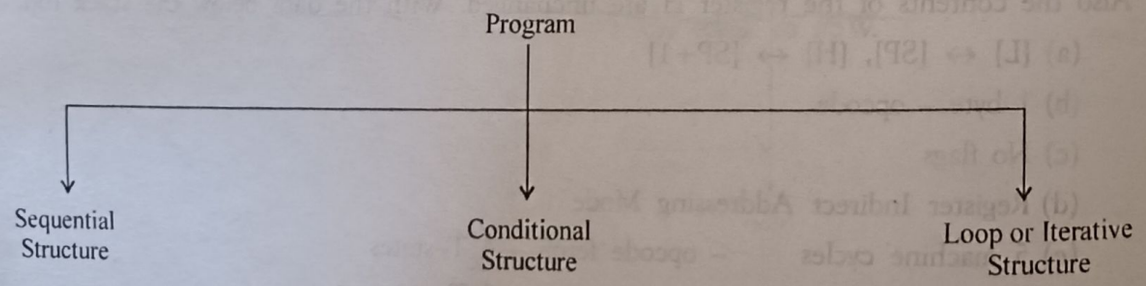


Fig. 4.6 : Basic Structures of Program

4.3.1 Sequential Structure

In this type of structure the execution of the instructions of the programs are performed one by one in a sequence in which they are written. The total sequence goes from top to bottom without any branching or looping.

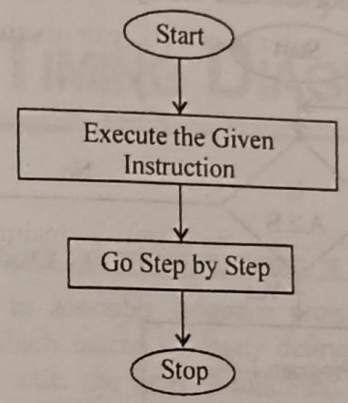


Fig. 4.7 : Flow Chart of Sequential Structure

4.3.2 Conditional Structure

In this program certain condition is checked and satisfied, may be result is zero, carry is coming or not, greater or smaller etc. Consider the two programs P1 and P2, if C = 1 is the given condition then P1 can be executed and if the condition is false P2 can be executed.

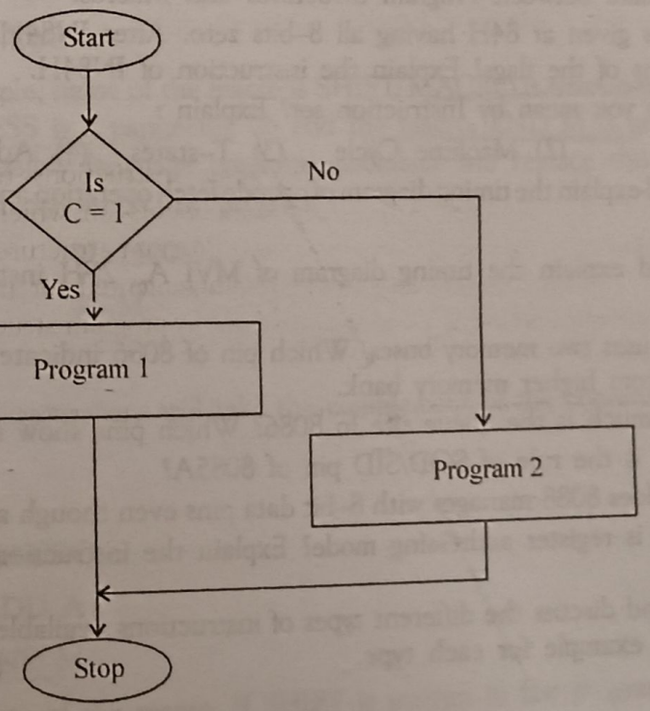


Fig. 4.8 : Flow Chart of Conditional Structure

4.3.3 Iterative Structure or Loop Structure

The type of program in which a condition is satisfied after checking and program control moves in a loop else it does not goes into a loop is called a loop structure or iterative structure. If A > B is the given condition then the program will be executed and if the condition is not satisfied, the processor will not execute the given program.

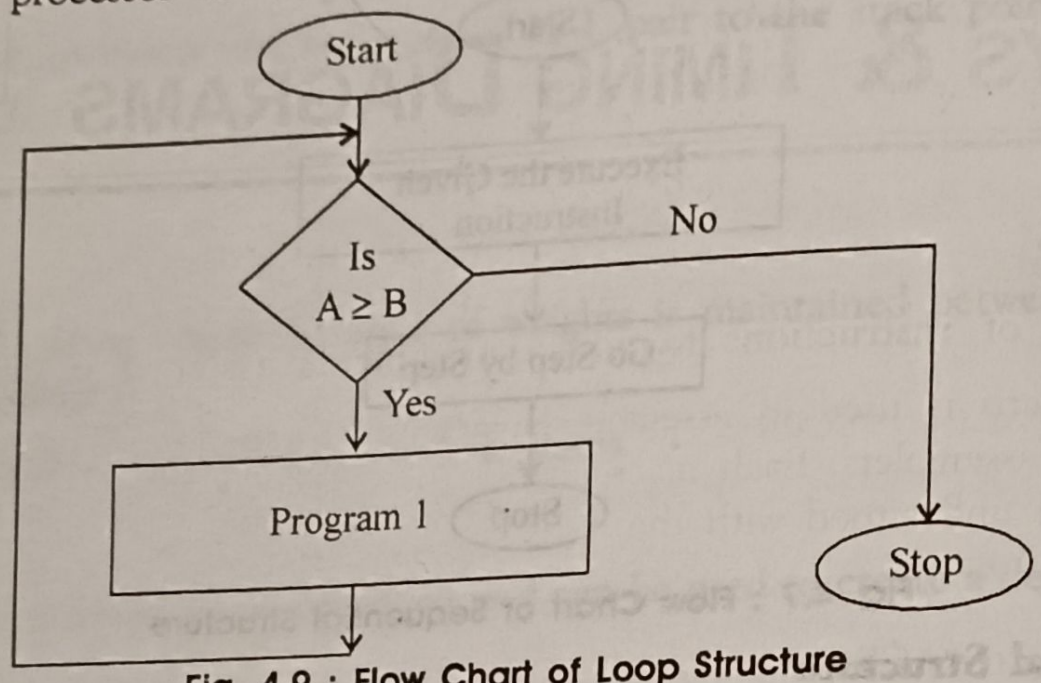


Fig. 4.9 : Flow Chart of Loop Structure

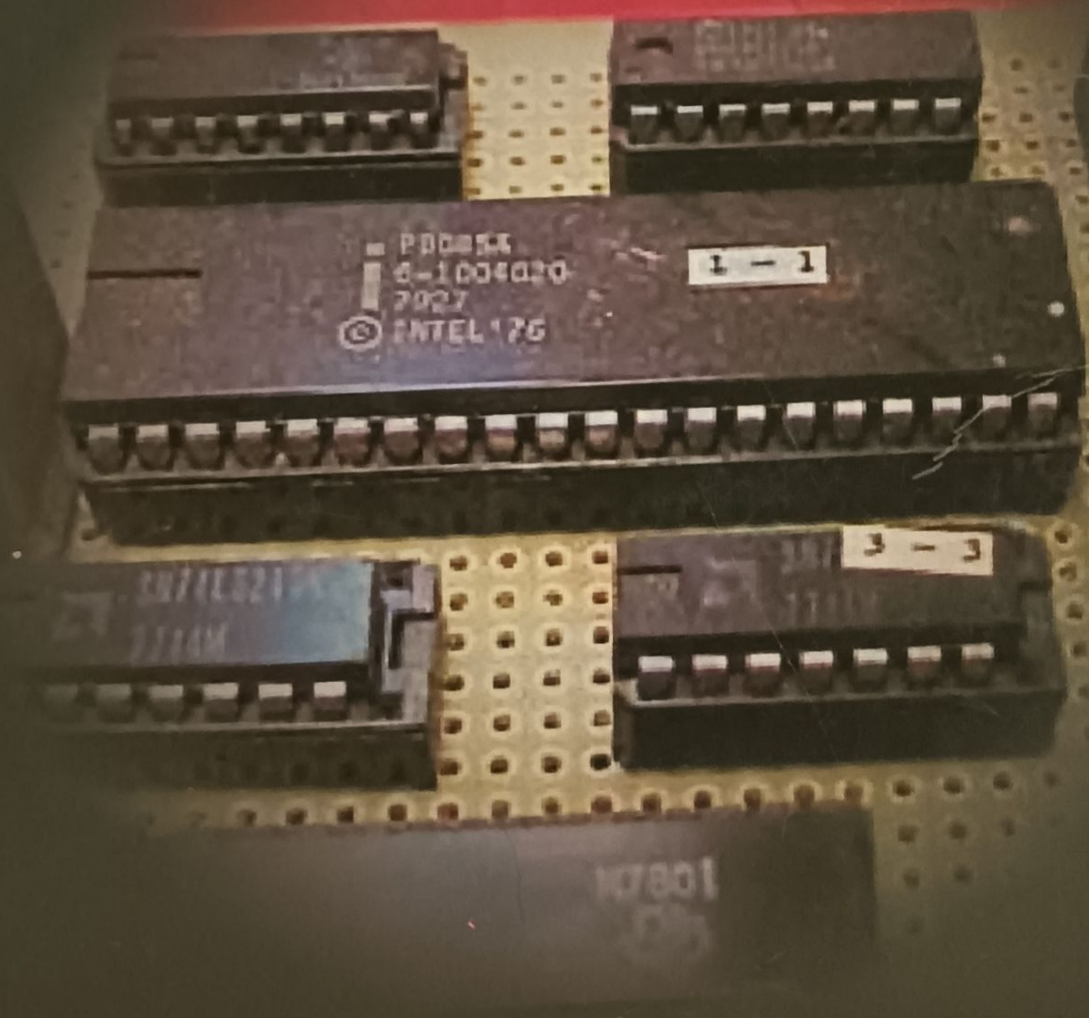
REVIEW QUESTIONS

1. Classify the instructions of 8085 according to the function.
2. Differentiate between Program Structures and Macros.
3. If data is given at 84H having all 8–bits zero. After IN84H what are the conditions of the flags? Explain the instruction of IN84H.
4. What do you mean by Instruction set? Explain :
   (1) Flag (2) Machine Cycle (3) T–states (4) Addressing Modes
5. Draw and explain the timing diagram of opcode fetch operation and memory read operation. **[ECE N.U. 2003]**
6. Draw and explain the timing diagram of MVI A_0, 2FH instruction. **[EIC (VI Sem) R.U. 2005]**
7. (a) 8086 uses two memory buses. Which pin of 8086 indicate that port of the data is coming from higher memory bank.
   (b) How much is the queue size in 8086? Which pins show the status of the queue?
   (c) What is the role of SOD/SID pin of 8085A?
   (d) How does 8088 manages with 8–bit data pins even though all registers are of 16–bit?
   (e) What is register addressing mode? Explain the instruction of 8085 in this mode. **[ECE N.U. 2001]**
8. Classify and discuss the different types of instructions available in 8085 microprocessor giving an example for each type. **[ECE (VI Sem) R.T.U. 2009]**

❑❑❑

N.K™

8085 MICROPROCESSOR & INTERFACES



RAHUL SRIVASTAVA
SHWETA SHARDA
MANISH SINGHAL

NEELKANTH PUBLISHERS (P) LTD.

8085

MICROPROCESSOR & INTERFACES

Rahul Srivastava
Senior Lecturer, Dept. of Electronics & Comm. Engg.
Arya Group of Colleges,
Jaipur

Shweta Sharda
Lecturer, Dept. of Electronics & Comm. Engg.
Arya College of Engineering & I.T.,
Jaipur

Manish Singhal
Lecturer, Dept. of Electronics & Comm. Engg.
Poornima College of Engineering,
Jaipur

NEELKANTH

NEELKANTH PUBLISHERS (P) LTD.