3 INSTRUCTION SET AND PROGRAMMING OF 8085

INTRODUCTION

As we all know, computers recognize and operate only with binary numbers. Each machine has its own instruction set based on the design of its CPU. Similarly for a microprocessor there exists its own instruction set. These instructions are in binary form and the language used is called machine language. Microprocessor design engineers select combinations of different bit patterns and assign a specific meaning to each combination by using electronic logic circuits. Each of these combinations is called as an instruction. Instructions are thus made up of one or more words. As we know a word is a group of bits. In the case of 8085 a word means an 8-bit group which is also called as a byte. A set of instructions designed into a machine makes up its machine language, which is specific to each machine. This chapter describes 8085 instruction set and its use in writing an assembly language programs.

2.1 INSTRUCTION CYCLE

Whenever any instruction is executed by a microprocessor number of different operations are to be carried out by the microprocessor. The clock achieves the overall control of the operations of microprocessor. Thus it may take several clock cycles to perform a particular operation. In the microprocessor terminology "the sub-division of an operation, which equals to one clock period is called as T-state". After defining the T-state we will define machine cycle and instruction cycle.

Machine Cycle

It is defined as "the time required to complete any operation, which is a sub-part of an instruction". The machine cycle may consist of number of T-states. For 8085 a machine cycle may consist of three to six T-states depending upon which machine cycle operation is being carried out. Following are the 8085–machine cycle operation.

No.	Machine Cycle Operation	Explanation of Operation
1.	Op-code Fetch	Fetches opcode from memory
2.	Memory read	reads data from memory
3.	Memory write	writes data into memory
4	Acknowledge	Acknowledges an interrupt.
7.	Halt.	Halts CPU while executing Halt instruction
8.	Hold.	Address data and control lines tri-stated for DMA operation.
9.	Reset	Reset operation after resetting.

Instruction Cycle

It is defined as "the time required to complete the execution of an instruction". An instruction cycle may consist of no. of, machine cycles. For 8085 an instruction cycle may consist one to five machine cycles.

Thus any of the instruction when executed will require no. of machine cycles to be performed and which requires several T-states. A diagrammatic representation of these concepts can be shown as below

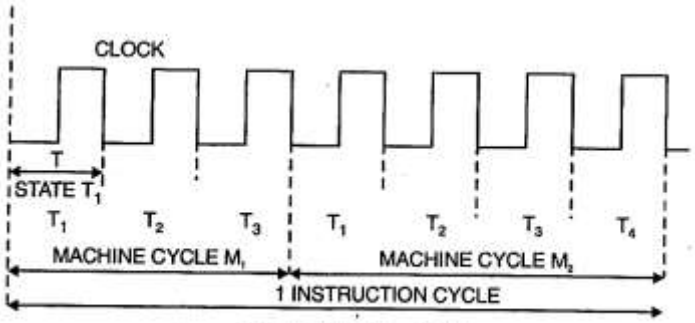


Fig. Instruction Cycle

Consider figure, which shows machine cycles T-states and Instruction cycle required for execution of a hypothetical instruction. From the diagram we see that the instruction requires two machine cycles M_1 and M_2. Machine cycle M_1 lasts for three T-states while machine cycle M_2 lasts for four T-states.

2.2 INSTRUCTION CLASSIFICATION OF 8085

Instruction set of 8085 can be classified into the following five categories depending upon their function.

1) Data transfer group
2) Arithmetic operations group
3) Logical operations group
4) Branching operations group
5) Machine control operations.

1) DATA TRANSFER INSTRUCTIONS

This group of instructions copies data from a location called source to another location called destination, without modifying the contents of the source. In technical manuals the term data transfer is used for this copying function. However the term 'transfer' is misleading; it creates the impression that the contents of source are destroyed when, in fact, the contents are retained without any modification.

Some of the various types of data transfer operations available in 8085 are as follows

	Types of Transfer	Example
1)	Between registers	Copy contents of register B to register E
2)	Specific data byte to register	Load register C with data byte or memory location 4AH
3)	Between memory location and a register	Copy data from memory Location 1500H to register D
4)	Between I/O device and accumulator (I/O operations)	Input from a keyboard to accumulator

The last type of operation namely I/O operation is sometimes grouped in separate group called I/O instructions.

2) ARITHMETIC GROUP INSTRUCTIONS

These instructions perform arithmetic operations such as addition, subtraction, increment, and decrement. 8-bit as well as 16-bit addition can be performed in 8085. 8-bit 2's complement subtraction is performed in 8085. Also 8-bit contents of register /memory location as well as 16-bit contents of register pair or stack pointer can be incremented or decremented using instructions in this group.

3) LOGICAL GROUP INSTRUCTIONS

These instructions perform following logical operations. In all the operations accumulator is one operand and where required second operand can be register content, memory content or 8 - bit data.

The results are stored in accumulator

a) AND operation
b) OR operation
c) XOR operation
d) Rotate
e) Compare
f) Complement or NOT operation.

4) BRANCHING GROUP INSTRUCTIONS

These instructions allow user to alter the sequence of execution of the program either conditionally or unconditionally.

Following are the types of Branch instructions available in 8085.

(a) Unconditional Jump
(b) Conditional Jump

In this case a jump is taken after checking for given condition.

(c) Call and Return instructions - These are the instructions used to enter into and return from a subroutine.
(d) Restart instructions - These are used to enter respective Interrupt service routine.

5) MACHINE CONTROL INSTRUCTIONS

These instructions are for following operations

a) Halt
b) Set interrupt mask
c) Read interrupt mask

d) No operation - (Do nothing)
e) Stack operations

2.3 INSTRUCTION FORMAT

As explained earlier an instruction is a combination of bit pattern. Normally each instruction can be viewed as a collection of two parts. One part, which gives the task to be performed, is rightly called as OPERATION CODE (Opcode) and the other part

gives the data to be operated on called as OPERAND. The operand can be specified in various ways.

Instructions for 8085 are of three formats.

i) One byte instructions ii) Two byte instructions iii) Three byte instructions

I) One Byte Instructions

These are the instructions, which include opcode and operand both in the same byte. In same cases the operand may be implicit. Such a type of instruction requires a single memory location. We will see examples of these types of instructions in next sections while learning the instruction set in detail, e.g. RRC, MOVB, C

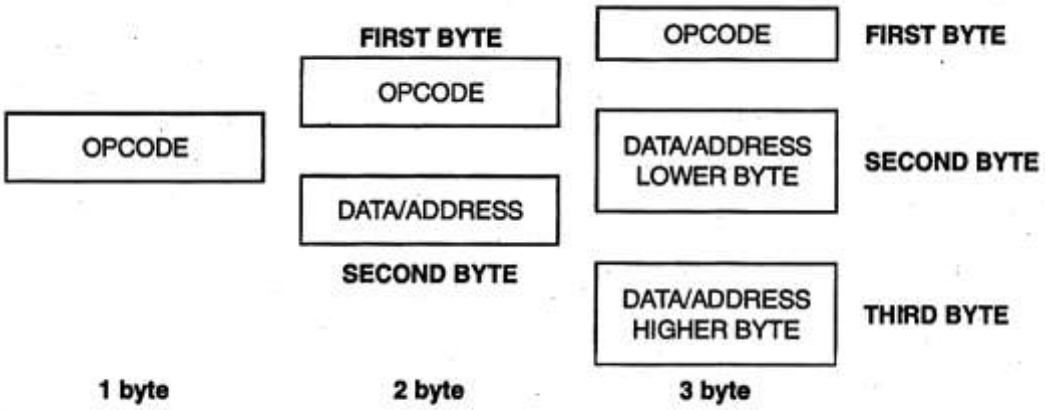


Fig. Instructions Formats

II) Two Byte Instructions

In a two-byte instruction, the first byte specifies opcode and the second byte specifies the operand. Such a type of instruction would require two memory locations to store in memory. Next few sections will explain about such instructions, e.g. MVI A,01H MVI A,(opcode-first byte) 01H(second byte)

III) Three Byte Instructions

In this type of instruction the first byte specifies the opcode as usual but the second and the third bytes together specify 16-bit address of operand or 16-bit data. But the important fact is the second byte is lower order byte and the third byte is higher order byte. Such an instruction would require three memory locations to store in memory. The examples and explanation for these type of instructions follows in next few sections, e.g. LXIH,2005H

LXIH,,(opcode-first byte[21]) 05H(second byte) 20H(third byte)

2.4 DATA TRANSFER INSTRUCTIONS

As seen earlier these instructions allow user to copy data from source to destination. Following are the instructions in this group.

1) MOVE INSTRUCTION

This instruction copies contents of source register to destination register. If one of the operands is a memory location then its address is specified in HL register pair. No flags are affected. The formats of these instructions are as follows.

Opcode	Operand	Bytes	Machine Cycles	T–states
MOV	Rd, Rs	1	1	4
MOV	M, Rs	1	2	7
MOV	Rd, M	1	2	7

Rs = Source location register. Rd = Destination location register.
Rd and Rs can be any of the following registers A, B, C, D, E, H, L.
M = Memory location whose address is given by contents of HL pair

Important Note:

1. The data in assembly language is always followed by 'H' e.g. 3AH 'H' stands for Hexadecimal number.
2. In this topic each instruction is specified with bytes, T states machine cycles this is for only information. It need not be remembered.

Examples:

MOV A, B ; Copy contents of B into A
MOV A, M ; Copy contents of memory location whose address is given by contents of HL register pair into accumulator.

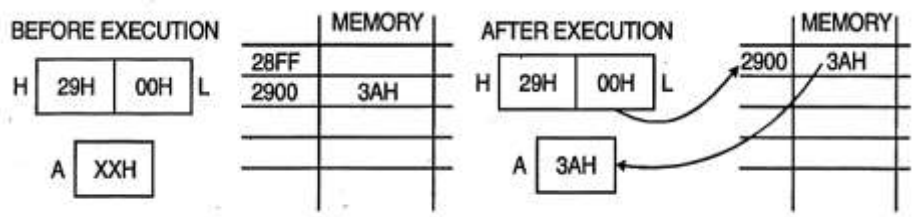


Fig. Idea of MOV A,M

2) MVI (MOVE IMMEDIATE) INSTRUCTION

This 2-byte instruction is used to store the 8-bit data, given in instruction itself, into the destination, which may be a register or memory location. The first byte of instruction gives the destination and second byte is the 8-bit data. No flags are affected.

Formats for these instructions are as follows.

Opcode	Operand	Bytes	Machine Cycles	T–states
MVI Rd,	data	2	2	7
MVI M,	data	2	3	10

Rd : destination register. Which can be any one of the following A, B, C, D, E, H, L
Data : 8-bit immediate data
M : memory location pointed by HL contents.

Examples:
MVI B, 3CH ; Store 3CH into register B.
MVI M, 50H ; Store 50H into memory location pointed by HL Contents.
MVI D, 5FH ; Store 5FH into register D.

3) LDAX-LOAD ACCUMULATOR INDIRECT

This is a single byte instruction. One of the operands is implicit and is accumulator. The other operand is a register pair. It is designated in instruction itself. The contents of the designated register pair point to a memory location. This instruction copies the contents of that memory location into the accumulator. The contents of either the register pair or the memory location are not altered. No flags are affected.

Opcode	Operand	Bytes	Machine Cycles	T–states
LDAX	rp	1	2	7

rp: register pair.
rp : B means BC register pair is used as pointer.
rp : D means DE register pair is used as pointer.

Example:
If BC register contents are 2500H i.e. B = 25H and C = 00H
and if LDAX B instruction is executed
then accumulator will be loaded with the contents of the memory location 2500 H.

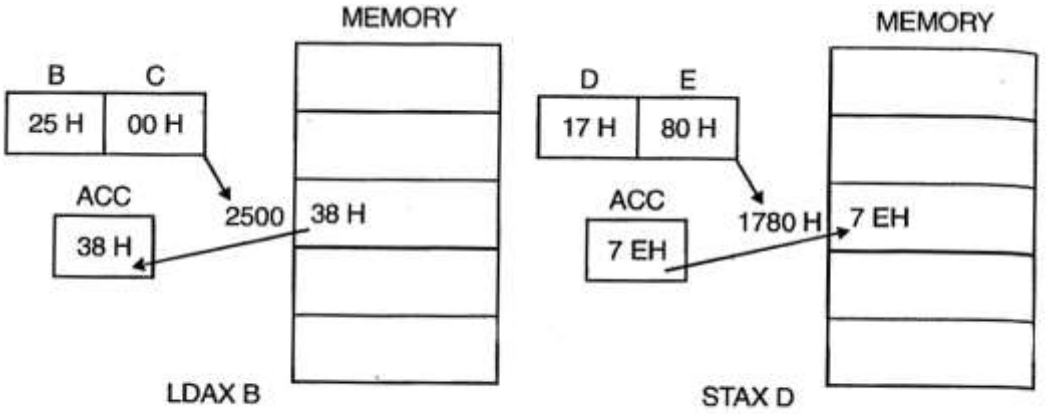


Fig. Illustration for LDAX & STAX instructions

4) STAX-STORE ACCUMULATOR INDIRECT

This is single byte instruction one of the operands is implicit and it is accumulator. The other operand is register pair. It is designated in instruction itself. The contents of the designated register pair point to a memory location. This instruction copies the contents of the accumulator into the memory location pointed by register pair. The contents of either register pair of the accumulator are not altered. No flags are affected.

Opcode	Operand	Bytes	Machine Cycles	T–states
STAX	rp	1	2	7

rp : register pair.

rp = B means BC register pair acts as pointer,.

rp = D means DE register pair acts as pointer.

Examples:

If contents of Accumulator are 7EH and D register contents 17H and E register contents 80H then after execution of the instruction STAX D the contents of memory location 1780 H will be 7EH.

5) LDA-LOAD ACCUMULATOR DIRECT

This is a 3 byte instruction, first byte gives opcode and the second and third provide a 16-bit address for a memory location. Second byte specifies lower order byte of provide a 16-bit address for a memory location. Second byte specifies lower order byte of address and third byte specifies the higher order byte of address. This instruction when executed copies contents of the memory location pointed by address given in instruction to the accumulator. Contents of source are not altered. No flags are affected.

Opcode	Operand	Bytes	Machine Cycles	T–states
LDA	addr	3	4	13

addr: 16-bit address.

Example:

If memory location 3780 H contains FEH and the instruction

LDA 3780H

is executed, then after the execution contents of accumulator will be FEH. The contents of memory location 3780 H are not altered. While writing this instruction in Hex code, we will write in following order

(opcode), 80H, 37H

OR (3A) , 80H, 37H where 3A is opcode of LDA

Where (opcode) is 8-bit machine code for the instruction?

6) STA-STORE ACCUMULATOR DIRECT

This is also a three-byte instruction. The first byte gives opcode and the second and third byte provides a 16-bit address which points to a memory location. As in the case of LDA instruction the second byte specifies lower order byte of the address and the third byte specifies higher order byte of the address. This instruction when executed

copies the contents of the accumulator into the pointed memory location. Accumulator contents are not destroyed. No flags are affected.

Opcode	Operand	Bytes	Machine Cycles	T–states
STA	addr	3	4	13

addr: 16-bit address Example:

Example:

If accumulator contents are 19H then executing the instruction

STA 2727H

Will copy 19H into the memory location 2727H. The contents of accumulator are not altered.

7) LXI- LOAD REGISTER PAIR IMMEDIATE

This is a 3-byte instruction first byte designates op-code and the register pair operand. The second and third byte specify a 16-bit data with usual convention of lower order byte first and higher order byte next. On execution this 16-bit data is copied into the designated register pair. No flags are affected.

Opcode	Operand	Bytes	Machine Cycles	T–states
LXIrp	Data	3	3	10

rp: register pair

rp can be any one register pair of the following.

Indication	Implied reg–pair
B	BC
D	DE
H	HL
SP	SP

Data: 16–bit data

Example:

LXI H, 2100 H will be written in the order (opcode), OOH, 21H. Upon execution H register will contain 21H and L register will contain will OOH.

(Instruction LXI in this case 'X' indicates there is a pair. Remember; if 'X' is present in any instruction that means there is a pair.)

8) LHLD - LOAD H AND L REGISTER DIRECT

This is also a 3-byte instruction. The first byte gives opcode. The second and third bytes specify a 16-bit address in usual convention of lower order byte first and higher order byte next. On execution the contents of memory location pointed by the specified address are copied into L register. The contents of the memory location next to specified location is copied into H register. Contents of these memory locations are not altered. No flags are affected.

Opcode	Operand	Bytes	Machine Cycles	T–states
LHLD	addr	3	5	16

addr: 16-bit address.

Example:

Assume memory location 2057H contains 55H and 2058H contains 25H then to transfer these to HL pair we will execute the instruction.

LHLD 2057H

Then after execution of above instruction

Register H will contain 25H and Register L will contain 55H

9) SHLD - STORE H AND L REGISTER DIRECT

This also is a 3-byte instruction. The first byte gives opcode. The second and third byte specify a 16-bit address in usual convention of lower byte first and higher order byte next. On execution the contents of L register are copied to the memory location pointed by the address specified in instruction and the contents of H register are copied to the memory location next to the specified location. Contents H and L registers are not altered. No flags are affected.

Opcode	Operand	Bytes	Machine Cycles	T–states
SHLD	addr	3	5	16

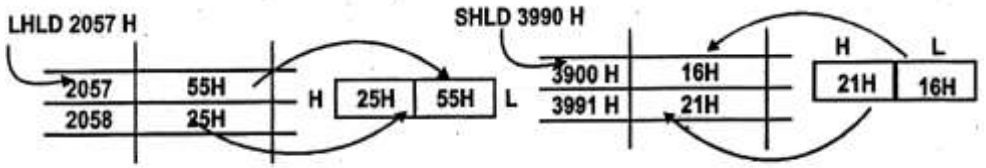


Fig. SHLD & LHLD Instructions

addr: 16–bit address.

Example :

Assume contents of HL register pair before execution as H = 21H, L = 16H

After execution of the instruction SHLD 3990H

We will have contents of memory location as refer figure

Memory location	Data
3990H	16H
3991H	21H

10) XCHG - EXCHANGE H AND L WITH D AND E

This is a single byte instruction. On execution of this instruction contents of H register are exchanged with contents of D register and contents of L register are exchanged with contents of E register. No flags are affected.

For this instruction the operands are implicit and no explicit operand is written.

Opcode	Operand	Bytes	Machine Cycles	T–states
XCHG	none	1	1	4

Example: Suppose contents of HL register pair and DE register are as follows.

Contents before execution of instruction:

Register	Data	Register	Data
H	15H	L	70H

D	25H	E	90H

But after executing the XCHG instruction the contents of these registers will be as follows. Contents of register after execution of XCHG instruction:

Register	Data	Register	Data
H	25H	L	90H
D	15H	E	70H

11) IN-INPUT 8-BIT DATA FROM AN INPUT PORT TO ACCUMULATOR.

This is the instruction which is included in the I/O instruction group. This is a two-byte instruction. First byte signifies the operation code for “input data from Port” operation. The second byte gives the 8-bit port address. When this instruction is executed, the microprocessor sends the 8-bit port address on lower address bus A0-A7. It also duplicates this address on higher address bus A8-A15. Any one of the set i.e. lower or higher address bus, can be decoded to enable the input port. The 8-bit data is then inputted from selected port into accumulator. No flags are affected.

In 8085 ports address can range from OOH to FFH. Which allows the user to have at the most 256 ports configured.

Opcode	Operand	Bytes	Machine Cycles	T–states
IN	Inport	2	3	10

Inport: 8-bit port address

Example: Consider the instruction IN 5BH

When this instruction is executed the microprocessor reads 8-bit data present at the input port 5BH.

12) OUT - OUTPUT 8-BIT DATA FROM ACCUMULATOR TO AN OUTPUT PORT

This is the other instruction, which is included in the I/O instructions group. This also is a two-byte instruction. The execution of this instruction is similar to the execution of IN instruction except that the direction of data transfer in this case is from microprocessor to output port. In other words the 8-bit data in accumulator is output on to the output port. As in the case of IN instruction the first byte signifies opcode and the second one informs the 8-bit output port address. No flags are affected.

Opcode	Operand	Bytes	Machine Cycles	T–states
OUT	Out part	2	3	10

Output port: 8-bit output port address.

Example: Consider the instruction OUT 27H

When this instruction is executed the contents of accumulator are output to the output port 27H.

This completes the discussion of Data transfer group of instructions. In the next two sections we will try to write simple 8085 assembly language programs. We will see how to write a correct and systematically written language program.

2.6 ARITHMATIC GROUP INSTRUCTIONS

As discussed earlier this group contains instructions, which allow arithmetic operations such as addition, subtraction, increment and decrement along with certain special instructions. Following are the instructions in this group.

1) ADD - ADD REGISTER TO ACCUMULATOR

This is a single byte instruction with operand placed along with opcode in the code byte. The operand can be a register or a memory location pointed HL register pair.

When this instruction is executed the contents of operand i.e. either register or memory location contents are added to the accumulator or the result is stored in accumulator. All flags are modified to reflect the result of addition. This addition is an 8-bit unsigned binary addition. The instruction has following formats.

Opcode	Operand	Bytes	Machine Cycles	T–states
ADD	reg	1	1	4
ADD	M	1	2	7

reg: any one of the following registers

M: memory location pointed by contents of the HL register pair.

Examples:

a) Consider the instruction ADD D

Let the contents of registers A and D before execution of above instruction be

D = 51H A = 47H

On execution of the ADD D instruction following addition is carried out

47 H =	0 1 0 0	0 1 1 1
+ 51 H =	0 1 0 1	0 0 0 1
98 H =	1 0 0 1	1 0 0 0

then sign flag = S = 1, (Since D7 bit of result is 1)

Zero flag = Z = 0 (Refer table of flags)

Aux Carry = AC = 0, Parity Flag = P = 0 Carry flag = C = 0

Thus contents of A,D and flags register, after the execution are as follows.

	S	Z	X AC	X P	X C
A = 98 H	1	0	0 0	0 0	0 0
D = 51 H			Flags =	80 H	

b) Consider the instruction ADD M

Let contents of A,H,L registers before the execution of above instruction be

A= 76 H, H = 25 H, L = 35 H

Let contents of memory location 2535 H be A2 H then after execution of ADD M A2 H will be added to 76 H and result will be stored in Accumulator. All flags will be

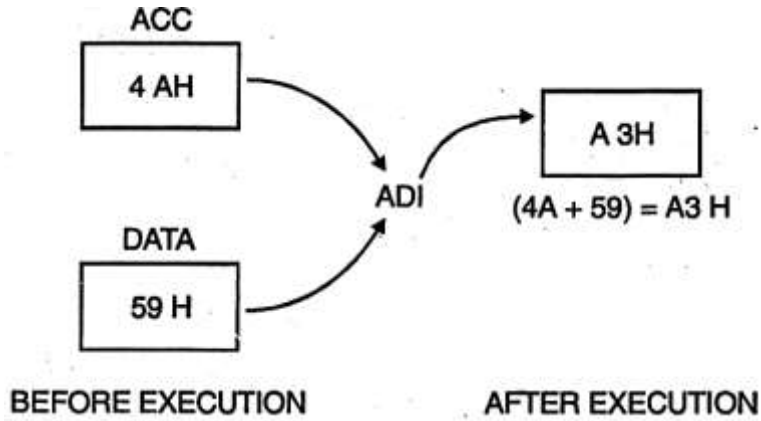
A	18 H	S	Z	X	AC	X	P	X	C
H	25 H	0	0	0	0	0	1	0	1
L	35 H	Flags = 05 H							

2) ADI - ADD IMMEDIATE TO ACCUMULATOR

This is a two by instruction. First byte gives opcode. The second byte is itself 8-bit data. This immediate data is added to accumulator contents. Result is stored in the accumulator and all flags are modified. This type of addressing mode is termed as an immediate addressing.

Opcode	Operand	Bytes	Machine Cycles	T–states
ADI	Data	2	2	7

data: 8-bit immediate data.



Example:

Let contents of A be 4A H . Consider the instruction ADI 59 H

After execution of above instruction the contents of accumulator will be A=A3 H and flags will be 95 H (after converting flag contents in to Hex).

S	Z	X	AC	X	P	X	C
1	0	0	1	0	1	0	0

3) ADC - ADD REGISTER TO ACCUMULATOR WITH CARRY

This also is a one-byte instruction. The operand, which is placed along with opcode in the code byte, can be either a register or the memory location pointed by HL pair. On execution of this instruction, contents of operand and carry flag are 9 added to accumulator and the result is stored in Accumulator. All flags are modified. This instruction is generally used for 16-bit addition.

Opcode	Operand	Bytes	Machine Cycles	T–states
ADC	reg	1	1	4
ADC	M	1	2	7

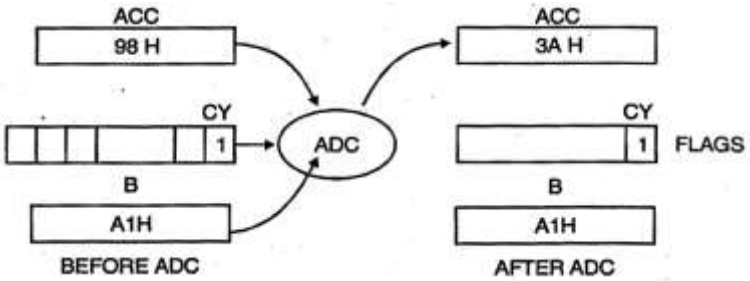
reg: any of the following registers B, C, D, E, H, L, A

M: memory location pointed by HL reg. pair

Example: Let contents of Accumulator, Carry flag, B register, before the execution of the instruction ADC B be

A = 98 H B = A1H Carry flag = 1

When ADC B is executed we get the result stored in accumulator. A = 3A H and carry flag = 1



4) ACI-ADD IMMEDIATE TO ACCUMULATOR WITH CARRY

This also is a two-byte immediate addressing mode instruction when it is executed, the 8-bit data (Second byte of instruction) is added along with carry to accumulator contents and the result is stored in accumulator. All flags are modified according to result.

Opcode	Operand	Bytes	Machine Cycles	T–states
ACI	data	2	2	7

data: 8-bit immediate data.

Example:

Let following be some of the register contents.

A=26 H Carry = 1

then after execution of the instruction

ACI 57 H

the contents will A = 7EH Carry flag = 0

5) SUB-SUBTRACT REGISTER OR MEMORY FROM ACCUMULATOR

This is also a single byte instruction when executed, works almost same as ADD instruction except that in this case memory or register contents are subtracted from accumulator and the contents of source are unaltered. All flags are modified to reflect result of subtraction. **CY flag is complimented after subtraction.**

Opcode	Operand	Bytes	Machine Cycles	T–states
SUB	reg	1	1	4
SUB	M	1	2	7

reg: any of the following registers B, C, D, E, H, L, A

M: Memory location pointed by HL pair

Example:

Consider the contents of Accumulator and register E before execution of the instruction SUBE

Contents before execution

A = 40H = 0100.0000 and E = 37H = 00110111

1' s complement = 1100 1000
2's complement = 1100 1000 +1 =1100 1001
A + E′ = 0100 0000 + 1100 1001 =1 0000 1001
= 09H
CY =1 after complementing CY =0

then when the instruction SUB E is executed a 2's complement subtraction is carried out as shown.

If result is positive the carry flag is reset and the accumulator contains the result. If results negative the carry flag is set and the accumulator contains 2's compliment of the magnitude of result. In this case contents of Accumulator and carry flag will be

A= 09 H Carry = 0

6) SUI-SUBTRACT IMMEDIATE FROM ACCUMULTOR

This instruction is also a two-byte instruction. It uses the immediate addressing mode. The first byte gives opcode and second byte gives immediate data. The immediate data is subtracted from the contents of accumulator and the result is stored in accumulator. All flags are modified according to the result. The subtraction is a 2′s complement subtraction and the significant of result is same as that for the SUB instruction.

Opcode	Operand	Bytes	Machine Cycles	T–states
SUI	data	2	2	7

data: 8-bit immediate data

Example:

Let the accumulator contents be 40 H. If we execute this instruction

SUI 37 H

then accumulator will contain 09 H with carry flag = 0 indicating that the result is positive. {Perform this subtraction by 2's coYnplement method)

7) SBB-SUBTRACT SOURCE AND BORROW FROM ACCUMULATOR

This also is a single byte instruction. It subtracts contents of source and borrow from the accumulator contents. It is a 2's complement subtraction. Result is stored in accumulator. All flags are modified according to the result. The source can be either any of the registers or any memory location. The memory location to be used, as source should be pointed by HL register pair. The carry flag acts as borrow.

Opcode	Operand	Bytes	Machine Cycles	T–states
SBB	reg	1	1	4
SBB	M	1	2	7

reg: any of the following registers B, C, D, E, H, L, A M: memory location pointed by HL pair

Example:

Let contents of Accumulator, D register and carry flag be

A = 37 H, D = 3F H and Carry = 1

After execution of the instruction SBB D

We get result in accumulator as A = F7 H with carry flag = 1

Carry flag set indicates that result in accumulator is negative and is in 2′s complement form.

8) SBI-SUBTRACT IMMEDIATE WITH BORROW

This is a two-byte instruction using immediate addressing mode. The first byte is opcode and second byte is 8-bit immediate data. The immediate data and borrow is subtracted from accumulator contents. It is a 2's complement subtraction. The result is stored in accumulator. The significance of result is same as that for SUI instruction. All flags are modified.

Opcode	Operand	Bytes	Machine Cycles	T–states
SBI	data	2	2	7

data: 8-bit immediate data

Example:

Consider the instruction SBI 20 H with initially set carry flag acting as borrow When executed it will subtract 21 H from accumulator contents and store the result in accumulator.

9) INR-INCREMENT CONTENTS OF REGISTER/MEMORY BY 1

This is a single byte instruction when executed the contents of operand are incremented by 1. All flags except carry flags are modified. The operand can be any of the register or a memory location pointed by HL register pair.

Opcode	Operand	Bytes	Machine Cycles	T–states
INR	reg	1	1	4
INR	M	1	3	10

reg: any of the following registers B, C, D, E, H, L, A

M: memory location pointed by HL pair

Example:

Let contents of E register be E = 2F H 0010 1111

If we execute the instruction INR E + 1

0011 0000 = 30H

then the modified E register and flags will be as follows

E = 30H = 0011 0 0 0 0 Flag = 14H

Flags

0	0	0	1	0	1	0	0 N/C
S	Z	X	AC	X	P	X	C

C = NC = last state

10) DCR-DECREMENT SOURCE BY 1

This is also a single byte instruction, when executed will decrement the contents of source by 1. All flags except carry are modified. The source can be any register or memory pointed by HL pair.

Opcode	Operand	Bytes	Machine Cycles	T–states
DCR	reg	1	1	4
DCR	M	1	3	10

reg: any of the following registers B, C,D, E, H, L, A

Example:

Contents of Accumulator before execution A 55 H

Contents of Accumulator and flag after execution A 54 H

Flags	0	0	X	0	X	0	X	NC
	S	Z		AC		P		C

11) INX-INCREMENT REGISTER PAIR BY 1

This is a single byte instruction. Which when executed increments the 16-bit contents of operand register pair by 1 No flags are affected. This is also called as extended increment.

Opcode	Operand	Bytes	Machine Cycles	T–states
INX	rp	1	1	6

rp : any of the following 16–bit operand (reg. pairs)

e.g if HL = 29FFH

then after INXH HL = 3000H (29FF+1 = 3000H)

(**Remember:** The difference between INR and INX: INR is used for register and INX is used for register pair. X represents Pair)

12) DCX-DECREMENT REGISTER PAIR BY 1

This also is a single byte extended decrement register. It decrements the contents of operand (16-bit) quantity by **1. No flags are affected.**

Opcode	Operand	Bytes	Machine Cycles	T–states
DCX	rp	1	1	6

rp: same significance as rp in INX instruction

e.g if BC = 40 01H

then after DCXB BC = 4000H (4001-1 =4000H)

13) DAD-ADD REGISTER PAIR TO H AND L REGISTERS

This is also a single byte instruction. It adds 16-bit contents of specified operand to 16-bit contents of H and L register pair. The result is stored in H and L register pair. Only carry flag is affected. No other flags are affected. Carry flag sets when result is greater than 16-bit number. This instruction is also called as double add instruction. Contents of operand are unaffected.

Opcode	Operand	Bytes	Machine Cycles	T–states
DAD	rp	1	3	10

rp: reg. pair operands same as rp for INX instruction

Example:

Assume contents of HL pair and SP as follows

H = 10 H L =10 H

S = 23H P = 23 H

If we execute the instruction DAD SP then modified contents of H, L and SP are as follows

H =33 H L = 33 H carry flag is reset

S = 23H P =23 H no other flags are affected

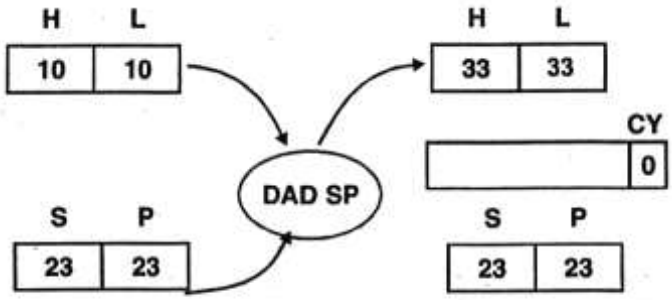


Fig. Idea of DAD instructions

14) DAA-DECIMAL ADJUST ACCUMULATOR

This is a special arithmetic instruction. It adjusts binary contents of accumulator to two 4-bit BCD digits. This is the only instruction which uses Auxiliary carry (AC) flag for code adjustment. The CPU uses the flag internally. All flags are modified to reflect the result of execution. The adjustment process is as follows. The important condition is that this instruction must always follow an addition instruction for two BCD numbers. Thus it adjusts the sum of two BCD numbers to BCD and does not convert a binary number to BCD. Also it cannot be used to adjust results after subtraction.

The adjustment procedure is as follows

1) Carry out BCD addition - previous instruction.
2) If only lower nibble of accumulator is greater than 9 or if AC flag is set add 06 to lower nibble.
3) If only higher nibble of accumulator is greater than 9 or if C flag is set add 06 to upper nibble.
4) If both upper and lower nibbles are greater than 9 or AC and C flags are set respectively then add 66 to the accumulator contents.

Opcode	Operand	Bytes	Machine Cycles	T–states
DAA	–	1	1	4

no emplicit operand

a) Let us add 12 BCD to 39 BCD

39 BCD =	0 0 1 1	1 0 0 1
+ 12 BCD =	0 0 0 1	0 0 1 0
51 BCD =	0 1 0 0	1 0 1 1

The binary quantity (sum) is 4 BH We views that conditions 1 and 2 applies so add 06 to lower nibble.

4 B =	0 1 0 0	1 0 1 1
+ 0 6 =	0 0 0 0	0 1 1 0
51 =	0 1 0 1	0 0 0 1

Thus accumulator contents are adjusted to BCD value.

b) Let us add 68 BCD to 85 BCD

68 BCD =	0 1 1 0	1 0 0 0
+ 85 BCD =	1 0 0 0	0 1 0 1
153 BCD =	1 1 1 0	1 1 0 1
	E	D

In the accumulator binary sum is EDH. We view that conditions 1 and 4 are fulfilled so we add 66 H to EDH.

EDH =	1 1 1 0	1 1 0 1
+ 66 H =	0 1 1 0	0 1 1 0
153 BCD =	1 0 1 0 1	0 0 1 1
	1 5	3 BCD

Thus accumulator contents are adjusted to BCD.
This completes the arithmetic group instructions.

2.8 LOGICAL GROUP INSTRUCTIONS

As seen earlier these instructions allow logical operations to be carried out. Logical operations such as ANDing, ORing, XORing, inverting etc. are allowed in 8085.
Following is detailed explanation for each of these instructions.

1) ANA-LOGICAL AND WITH ACCUMULATOR

This is a single byte instruction, which carries out logical ANDing of the contents of operand with the contents of Accumulator. The result is stored in accumulator itself.
Logical operations on 8-bit quantities are defined as 8 operations on respective bits. These 8 operations are independent.
S, Z, P flags are modified to reflect the result carry flag is reset and AC flag in set.

Opcode	Operand	Bytes	Machine Cycles	T–states
ANA	reg	1	1	4
ANA	M	1	2	7

reg: any of the following registers B, C, D, E, H, L, A
M: memory location pointed by HL pair.

Example:

The contents of accumulator and the register D are 54 H and 82 H, respectively.
If we execute the instruction

ANA D

The accumulator contents will be as follows:

A	= 54 H	0 1 0 1	0 1 0 0
AND			
D	= 82 H	1 0 0 0	0 0 1 0
Acc = A.D	=	0 0 0 0	0 0 0 0 = 00H

Thus accumulator contents after ANDing are 00H. A_0 is ANDed with D_0 and so on upto A7 ANDed with D_7.
This instruction is used to mask required bits during programming.

2) ORA-LOGICALLY OR WITH ACCUMULATOR

This single byte instruction carries out logical ORing of the contents of operand and contents of accumulator. The result is stored in accumulator itself. S, Z, and P flags are modified according to the result and AC and CY flags are reset.

Opcode	Operand	Bytes	Machine Cycles	T–states
ORA	reg	1	1	4
ORA	M	1	2	7

reg.: any of the following registers B, C, ,D E, H, L, A

M: memory location pointed by HL pair.

Example: Consider ORA B

Contents before execution

A B A H

B 1 1 H

Contents after execution

A B B H

B 1 1 H

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

BA H	=	1011	1010	
11H	=	0001	0001	
ORAB	=	1011	1011	= BBH

Flag = 10000 0100 = 84 H

3) XRA- EXCLUSIVE-OR WITH ACCUMULATOR

This is a one-byte instruction. It makes EX-Oring the operand contents with Accumulator. The result is stored in Accumulator S, Z, P flags are modified according to result. CY and AC flags are reset.

Opcode	Operand	Bytes	Machine Cycles	T–states
XRA	reg	1	1	4
XRA	M	1	2	7

reg.: any of the following registers B, C, D, E, H, L, A

M: memory location pointed by HL pair

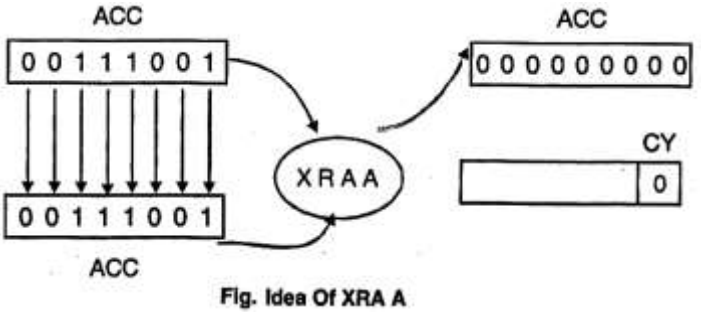


Fig. Idea Of XRA A

Example: i) Consider XRAB

Contents before execution

A AAH

B 55H

Contents after execution

A FF H

H 55H

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

AA H	=	1010	1010	
55 H	=	0101	0101	
XRAB	=	1111	1111	= FFH

Flag = 10000 0100 = 84 H

ii) Consider XRAA as shown in fig.(2.9) if A= 39H then after XRAA it makes accumulator clear

4) CMP-COMPARE WITH ACCUMULATOR

This is a single byte instruction. This instruction compares the contents of the operand (reg/memory) with the contents of accumulator. Both contents are preserved and a result of comparison is shown by setting the flags as below.
a) If [A] < [reg/memory] then Carry flag is set.
b) If [A] = [reg/memory] then Zero flag is set.
c) If [A] > [reg/memory] then both Carry and Zero flags are reset.
Note: [A] means contents of accumulator thus "[ ]" sign indicates "contents of ". In addition to Z and carry the other flags S, P, AC are also modified.

Opcode	Operand	Bytes	Machine Cycles	T–states
CMP	reg	1	1	4
CMP	M	1	2	7

reg.: any of the following registers B, C,D E, H, L, A
M: memory location pointed by HL pair.
Example: Consider the instruction CMP L
We will view different cases for different values of A and L

After execution				
Case No.	**Contents of A**	**Contents of L**	**Carry flag**	**Zero flag**
1	15 H	57 H	1	0
2	25 H	25 H	0	1
3	35 H	17 H	0	0

5) ANI-AND IMMEDIATE WITH ACCUMULATOR

This is a two byte instruction. It uses immediate addressing mode. The second byte acts as an 8-bit immediate data. It is logically ANDed with the contents of accumulator. The result is stored in accumulator S, Z and P flags are modified to reflect the result in the accumulator. Carry flag is reset, AC flag is set.

Opcode	Operand	Bytes	Machine Cycles	T–states
ANI	data	2	2	7

data: 8-bit immediate data.

Example: The instruction ANI 3FH when executed, will store the result of ANDing 3FH with accumulator in accumulator.

6) ORI-OR IMMEDIATE WITH ACCUMULATOR

This also is a two byte instruction. It uses immediate addressing mode. The second byte is immediate 8-bit data. This data is logically ORed with the contents of the accumulator. The result is stored in the accumulator itself. S, Z, and P flags are modified to refleGt the result in accumulator. Carry and AC flags are reset.

Opcode	Operand	Bytes	Machine Cycles	T–states
ORI	data	2	2	7

data: 8-bit immediate data.

Example: The instruction ORI4CH when executed, will OR the contents of accumulator with 4CH and will store the result in accumulator.

7) XRI-XOR IMMEDIATE WITH ACCUMULATOR

This is again a 2-byte instruction using immediate addressing mode. The second byte which is immediate data is XORed with accumulator contents and result is stored in accumulator itself S, Z and P flags are modified to reflect the result in accumulator, carry and AC flags are reset.

Opcode	Operand	Bytes	Machine Cycles	T–states
XRI	data	2	2	7

data: 8-bit immediate data.

Example: The execution of the instruction XRI FFH will store the result in accumulator, the result obtained by XORing accumulator contents with FFH.

8) CPLCOMPARE IMMEDIATE WITH ACCUMULATOR

This instruction is a two byte instruction utilizing immediate addressing mode. The execution of this instruction is very much similar to CMP instruction. The only difference being that in CPI instruction the immediate data is compared with accumulator.

S, P, AC flags are also modified in addition to Z and carry flag which reflect the result of comparison.

Opcode	Operand	Bytes	Machine Cycles	T–states
CPI	data	2	2	7

9) CMA-COMPLEMENT THE ACCUMULATOR

This is a single byte instruction. When executed this instruction complements the accumulator contents. The result is stored in accumulator itself. No flags are affected.

Opcode	Operand	Bytes	Machine Cycles	T–states
CM	–	1	1	4

Examples: a) Consider the accumulator containing 57 H

A = 0 1 0 1 0 1 1 1

On execution of the instruction CMA the accumulator contents will change to

A = 1 0 1 0 1 0 0 0

i.e. A = A8 H

b) Write a program to find l's complement of the number stored in memory location 65B0 H and store the result in memory location 6651H.

Program:

Label	Mnemonic	Opreand		Comments
	LDA	65B0 H	;	Load data in Acc
	CMA	–	;	Complement Acc
	STA	6651 H	;	Store result in memory
	HLT	–	;	Stop processing

10) CMC-COMPLEMENT CARRY

This also is a single byte instruction. When executed it complements the carry flag. No other flags are affected.

Opcode	Operand	Bytes	Machine Cycles	T–states
CMC	–	1	1	4

11) STC-SET CARRY

This is also a one-byte instruction. When executed it sets the carry flag to 1. No other flags are affected.

Opcode	Operand	Bytes	Machine Cycles	T–states
STC	–	1	1	4

12) RLC-ROTATE ACCUMULATOR LEFT

This is a single byte instruction. It is used to rotate each binary bit in accumulator to left by one position. Bit D7 is placed in bit position Do as well as in the carry flag. No flags other than carry flag are affected.

Opcode	Operand	Bytes	Machine Cycles	T–states
RLC	–	1	1	4

This operation can be represented as follows: CARRY

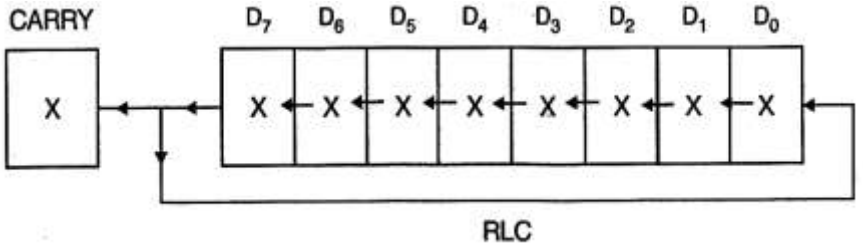
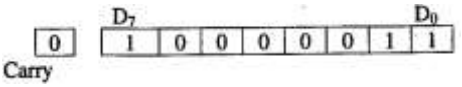


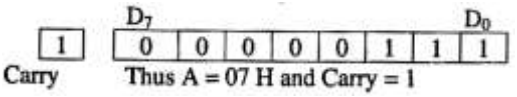
Fig. RLC INSTRUCTION

Example:

Let contents of Accumulator be 83 H Let Carry = 0



If we execute the instruction RLC then the accumulator and carry will be as follows:



13) RRC-ROTATE ACCUMULATOR RIGHT

This is also a single byte rotate instruction. It is used to rotate each binary bit in accumulator to right by 1 position. Bit D_0 is placed in bit position D_7 as well as in carry flag. No other flags are modified.

Opcode	Operand	Bytes	Machine Cycles	T–states
RRC	–	1	1	4

Example: This operation can be represented as follows

Let contents of accumulator be 83 H. Let carry be 0

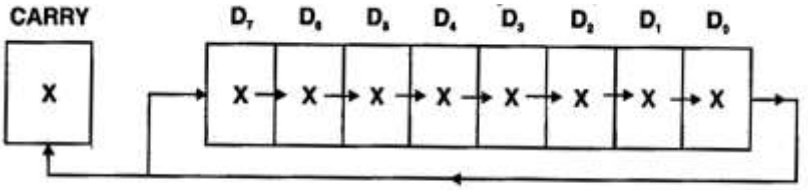
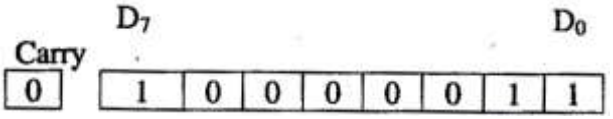
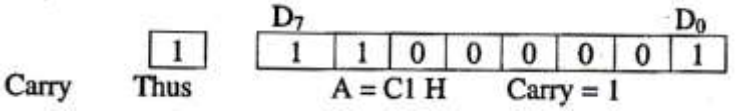


Fig. RRC INSTRUCTION



If we execute the instruction RRC then contents of accumulator and carry will be as follows



14) RAL-ROTATE ACCUMULATOR LEFT THROUGH CARRY

This instruction is also a one byte instruction. It rotates ech binary bit of accumulator left by one bit position through carry flag. No other flags are modified. D_7 is moved to carry and carry is moved to D_0.

Opcode	Operand	Bytes	Machine Cycles	T–states
RAL	–	1	1	4

This instruction can be represented as follows:

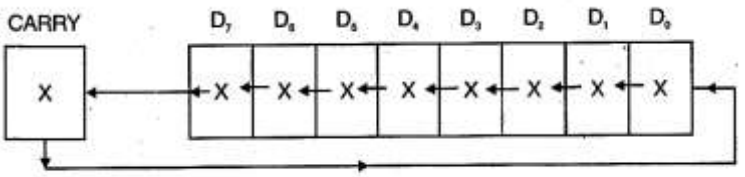


Fig. RAL INSTRUCTION

Example:

Let contents of accumulator be 83 H. Let carry be 0.

Carry		D_7							D_0
0		1	0	0	0	0	0	1	1

After execution of the RAL instruction.

We get following in Accumulator and carry flag.

Carry		D_7							D_0
1		0	0	0	0	0	1	1	0

i.e. A = 06 H and Carry = 1

15) RAR-ROTATE ACCUMULATOR RIGHT THROUGH CARRY

This is an instruction, which rotates the accumulator contents to 1 bit position right the carry i.e.

D_0 goes to carry and carry goes to D_0. No other flags are affected.

Opcode	Operand	Bytes	Machine Cycles	T–states
RAR	–	1	1	4

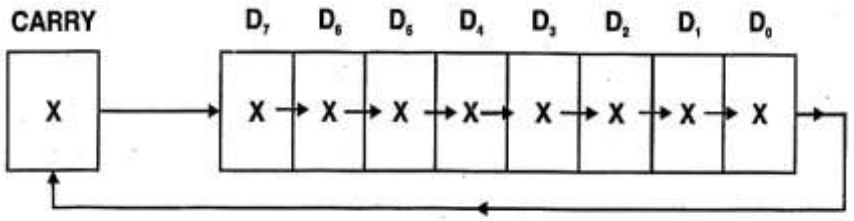


Fig. RAR INSTRUCTION

This instruction can be represented as follows.

Example :

Let A = 83 H Carry = 0

Carry	D7	D_0
0	1 0 0 0 0 0 1 1	

On execution of RAR

Carry	D7	D_0
0	0 1 0 0 0 0 0 1	

A = 41 H Carry =1

This completes the discussion of Logical group instructions.

These instructions CMC, STC, CMA are sometimes called special instructions.

2.9 BRANCHING CONTROL GROUP OF INSTRUCTIONS -JUMP INSTRUCTIONS

As we have seen conditional jump and unconditional jump are the branching instructions. These are as follows.

1) UNCONDITIONAL JUMP - JMP

This instruction is a three-byte instruction. First byte is opcode, second byte is lower order address for branching and the third byte gives higher order address byte for branching. When this instruction is executed the program control is unconditionally transferred to the branch address given in instruction. Such an instruction allows user to set up unconditional loops. No flags are affected.

Opcode	Operand	Bytes	Machine Cycles	T–states
JMP	addr	3	3	10

Addr. 16 bit branch address

Example

The instruction JMP 3500H when executed will transfer the program control to the memory location 3500 H.

2) CONDITIONAL JUMP INSTRUCTIONS

These are also 3 byte Jump instructions. The second and third byte gives branching address with same significance as that for in conditional jump instruction. The only difference between conditional and unconditional branch is that conditional branch instruction first checks a certain condition. If condition is true then only branching takes place else program continues in same sequence. No flags are affected. Following are the conditional jump instruction.

Opcode	Description	Required status of flag for jump to be taken
JC	Jump on Carry	CY = 1
JNC	Jump on No Carry	CY = 0
JP	Jump on Positive	S = 0
JM	Jump on Minus	S = 1
JPE	Jump on parity even	P=1
JPO	Jump on odd parity	P = 0
JZ	Jump on Zero	Z=1
JNZ	Jump on no Zero	Z = 0

For all these instructions operand is 16–bit branch address.

2.11 STACK

The stack in an 8085 based microprocessor system can be described as a set of R/W memory locations, specified by the program in the main program. These memory locations are used to store binary information (bytes) temporarily during the execution of a program.

The beginning of the stack is defined in the program by using a LXI SP instruction, which loads 16-bit memory address in stack pointer register of the microprocessor. In 8085 when we initialize stack pointer then storing of data bytes begins from a location with a address one less than the SP contents. And as we go on storing more and more data SP goes decrementing. Thus the stack grows from higher address to lower address. Therefore normally while writing 8085 programs users initialize SP at the highest available user R/W memory locations.

The size of stack is thus only limited by the available memory.

In 8085 we have following instruction to store the data and retrieve it back from stack.

a) PUSH To store operand contents on stack.

b) POP To load from stack into operand.

We will discuss these instructions in detail in forthcoming section.

One of the important uses of stack is done while executing a subroutine, but what exactly is a subroutine? Let us now see what does a subroutine exactly means and how it is executed.

2.12 SUBROUTINE

A subroutine is a group of instructions written separately from the main program to perform a function that occurs repeatedly in the main program. To avoid repetition of the same delay instructions, the subroutine technique is used. Delay instructions are written once only separate from the main program. The main program when required then calls these.

The 8085 microprocessor has two types of instructions to implement subroutines. '

a) CALL instructions: These are conditional and unconditional calls. These instructions are used to call a subroutine.
b) Return instructions: These are conditional and unconditional returns. These instructions are used to return from a subroutine to main program.

The call instruction is written in the main program at a location where you wish to execute the subroutine. The return instruction is written at the end of the subroutine indicating end of subroutine.

When CALL instruction is executed the contents of program counter i.e. address of instruction following CALL are stored in the stack and the program control is transferred to subroutine. On completing the execution of subroutine the RET instruction is executed which loads back the stack contents i.e. address of the instruction following CALL instruction, in program counter. Thus the main program execution is resumed.

In the next section we will see subroutine control instructions in the branch control group.

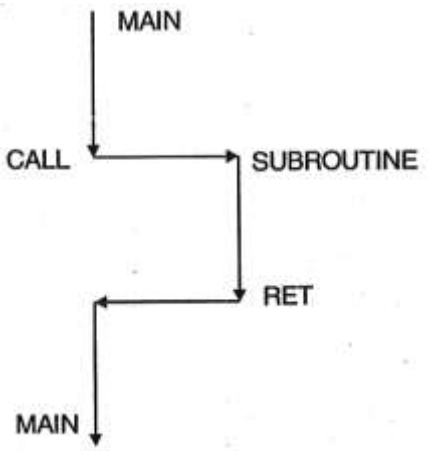


Fig. Idea of Subroutine

2.13 BRANCHING CONTROL GROUP INSTRUCTIONS

1) UNCONDITIONAL CALL - CALL

This is a 3-byte instruction that transfers the program sequence to a subroutine address. First byte is opcode and second and third bytes give the subroutine address. On execution the address of the instruction following CALL is stored in stack. It decrements SP by two. It then jumps unconditionally to the subroutine address. This instruction is accompanied by a return type instruction at the end of subroutine. No flags are affected.

Opcode	Operand	Bytes
CALL	addr	3

addr : 16 bit subroutine address

Example:

Consider the following program sequence starting from memory 2000 H. Let there be a subroutine from 3000 H

Main	Subroutine
2000 CALL 3000 H	3000 MOV C,A
2003 MOV A,C	3001 IN 23 H
	3003 RET

On executing CALL 3000 H the address of the MOV A,C instruction i.e. 2003 H is stored in stack and program control transfers to address 3000 H. On executing RET, stack contents are loaded in PC and main program execution resumes,

2) CONDITIONAL CALL INSTRUCTIONS

These are also 3-byte call instructions but are conditional meaning that they check a certain condition and the subroutine is called only when the condition is true. No flags are affected.

Opcode	Description	Required status of flags for Call to be executed
CC	Call on Carry	CY = 1
CNC	Call on No Carry	CY = 0
CP	Call on Positive	S = 0
CM	Call on Minus	S = 1
CPE	Call on Parity Even	P = 1
CPO	Call on Odd Parity	P = 0
CZ	Call on Zero	Z = 1
CNZ	Call on no Zero	Z = 0

3) UNCONDITIONAL RETURN - RET

As seen earlier a return instruction indicates end of subroutine and transfers the control back to main program. This is a single byte instruction. On execution it loads two byte from stack i.e. address of instruction next to CALL into PC and increments SP by two. No flags are affected.

Opcode	Operand	Bytes	Machine Cycles	T–states
RET	–––	1	3	10

4) CONDITIONAL RETURN INSTRUCTIONS

These are also one-byte instructions indicating end of subroutine on checking a certain condition and finding it true. Thus the control is transferred to main program in the same manner as for unconditional return only when condition is true else the subroutine execution is maintained. No flags are affected.

Opcode	Description	Required status of flags for Call to be executed
RC	Return on Carry	CY = 1
RNC	Return on No Carry	CY = 0
RP	Return on Positive	S = 0
RM	Return on Minus	S = 1
RPE	Return on Parity Even	P = 1
RPE	Return on Odd Parity	P = 0
RZ	Return on Zero	Z = 1
RNZ	Return on no Zero	Z = 0

5) RESTART INSTRUCTIONS - RST

We have seen the use and functioning of interrupts. We know that on giving the 8085 an interruption the pin INTR (Pin No. 10) the microprocessor fetches the Interrupt service Routine (ISR) branch address from data bus. For this we should externally provide the instruction. RST instructions are one of the instructions, which can be used to transfer the control to ISR. These are 8 different RST instructions RSTO to RST7. These are all one byte instruction. In execution they are similar to call instruction with a predefined subroutine address obtained from the number associated with RST.

As a rule if n is the number associated with RST then the subroutine or ISR address is 8n. No flags are affected.

These instructions can be used in software to generate interrupts thus these are also called software interrupts.

Opcode	Operand	Bytes	Machine Cycles	T–states
RST	n	1	3	12

i = any no. from 0 to 7.This completes the subroutine control instructions.

2.14 MACHINE CONTROL GROUP OF INSTRUCTIONS

A) STACK OPERATION INSTRUCTIONS

We have seen the use of stack, we now see the instructions, which allow one to store data on stack and retrieve it back. We will also see some data transfer instructions involving stack pointer (SP).

1) PUSH - PUSH REGISTER PAIR ON STACK

This is a single byte instruction. The contents of the register pair specified in the operand are copied into the stack in the following sequence.

a) The stack pointer is decremented and the contents of higher order register in pair (such as B in BC pair, D in DE pair) are copied on stack.

b) The stack pointer is decremented again and contents of lower order register are copied on the stack. No flags are modified Contents of register pair are unchanged.

Opcode	Operand	Bytes	Machine Cycles	T–states
PUSH	rp	1	3	12

rp : register pair any one of the following.

rp	Pair Name	
	High Byte	Low Byte
B	B	C
D	D	E
H	H	L
PSW	A	F

A : Accumulator F : Flag register PSW : Program status word.

Example: PUSH D

Will push contents of DE pair

Let D = 15 H & E = 23 H Let SP = 2300 H

Then after executing PUSH D we will get following contents in SP and stack.

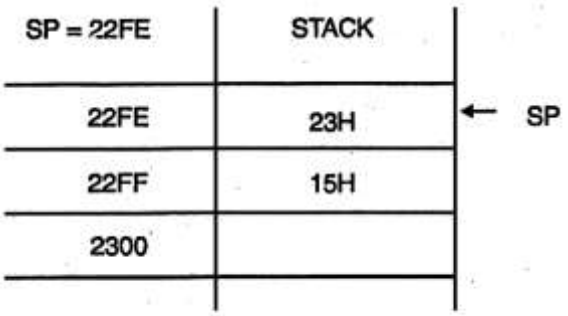


Fig. Push Instructions

2) POP - POP OFF STACK TO REGISTER PAIR

This is a single byte instruction. On execution copies two top bytes on stack to designated register pair in operand. The execution follows the sequence given below.

a) Contents of top most location of stack called stack top are copied into lower register (such as C in BC etc) of the pair. The SP is incremented by 1.

b) Contents of the stack location pointed by SP are copied into higher register of the pair. The stack pointer SP is incremented by 1.

No flags are affected. Contents of stack are unchanged.

Opcode	Operand	Bytes	Machine Cycles	T–states
POP	rp	1	3	10

Example:

Consider SP = 22FE H with following contents stored on stack.

On execution of instruction POP H the contents of H, L, SP will be as shown in figure.

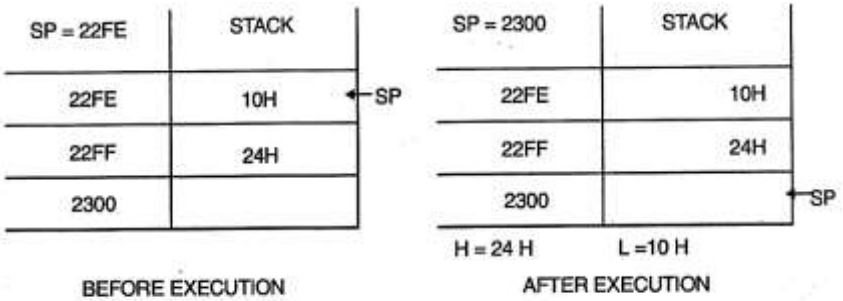


Fig. POP Instruction

3) XTHL - EXCHANGE H AND L WITH TOP OF STACK

This is a single byte instruction. On execution the contents of L register are exchange with contents of stack top byte i.e. Contents at memory location pointed be SP. Also contents of H register are exchanged with the contents of memory location pointed by SP + 1. Contents of SP are not altered. No flags are affected.

Opcode	Operand	Bytes	Machine Cycles	T–states
XTHL	––	1	5	16

Example:

Consider following stack contents and HL pair contents.

		Stack	
H = 26 H	L = 15 H	AB H	SP
		CDH	

On execution of the instruction XTHL the contents of HL pair and stack are as follows.

		Stack	
H = CD H	L = ABH	15 H	SP
		26 H	

4) SPHL - COPY H AND L REGISTER TO SP

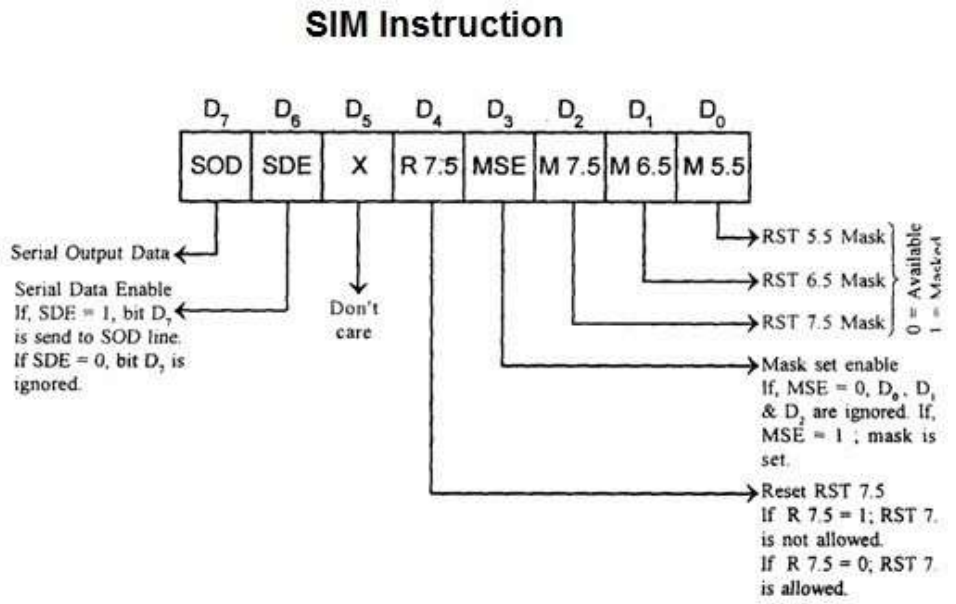
This is a single byte instruction which copies contents of L register into lower byte of SP and contents of H register into higher byte of SP. The contents of H and L registers are unaffected. No flags are affected.

Opcode	Operand	Bytes	Machine Cycles	T–states
SPHL	––	1	1	6

B) INTERRUPT CONTROL OPERATIONS

1) SIM- Set Interrupt Mask

Here is the format of the SIM instruction. The SIM instruction uses the data in the accumulator as follows:



D7-D6 - The left two bits are related to the serial interface. When D6 (SDE-Serial Data Enable) is 1, then whatever is in D7 (SOD-Serial Data Output) is written to the serial data output (pin 4 of the 8085). If D6=0, nothing is written. This allows a SIM instruction to beexecuted altering interrupt masks without affecting serial data.

Bit D5 is not used.

Bit D4 (R 7.5-Reset RST 7.5) This bit allows the SIM instruction to reset the interrupt pending flag indicated by bit D6 in the RIM instruction layout. The 7.5 interrupt can indicatethat it is pending via the RIM instruction even though it is masked off. This bit allows that pending request to be reset.

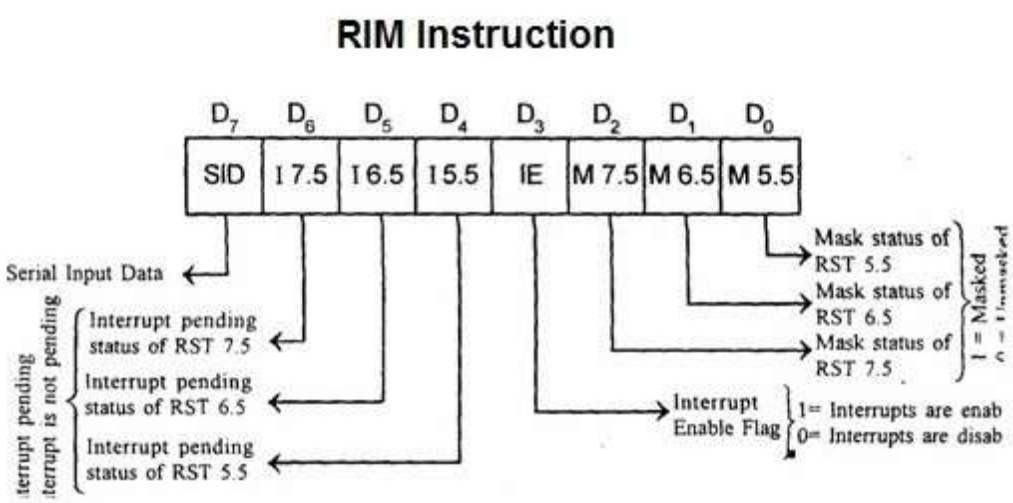
Bit D3 (MSE-Mask Set Enable) is like SDE -- it indicates whether the lower three bits (D2- D0) are ignored or not. This allows the serial data output to occur without affecting the interrupt masks. If a SIM is executed with this bit low, the condition of the mask bits will notchange. If a SIM is executed with this bit set high, the mask bits will be set according to the lower three bits of the accumulator.

Bits D2-D0 (RST 7.5 Mask, RST .5 Mask, RST 5.5 Mask) These are the interrupt masks for the 8085 interrupts 7.5, 6.5, and 5.5. If the corresponding bit is 0, the interrupt is enabled. Ifthe bit is 1, the interrupt is masked (ignored).

Here is the format of the RIM instruction. The RIM instruction reads the following bits Intothe accumulator:

2) RIM-Read Interrupt Mask

Here is the format of the RIM instruction.



Bit D7 (SID-Serial Input Data) This is the input pin of the serial data interface which is connected to pin 5 of the 8085, and indicates the high/low status of that pin.

Bits D6-D4 (I 7.5, I 6.5, I 5.5) These bits indicates that an interrupt is pending for these three 8085 interrupts 7.5, 6.5, and 5.5. If interrupts 5.5 or 6.5 have been masked off by bits D0 or D1, bits D4 and D5 will not be set. Bit D6, which corresponds to the 7.5 interrupt, will be set on to indicate that an interrupt 7.5 was requested, even if it was masked off.

Bit D3 (IE-Interrupt Enable) This bit indicates whether interrupts are enabled (1) using the EI(Enable Interrupts) instruction, or disabled (0) using the DI (Disable Interrupts) instruction.

Bits 2-D0 (M 7.5, M6.5, M5.5) Mask status of interrupts 7.5, 6.5, and 5.5. Corresponds tobits D2-D0 of the SIM instruction. 1 if masked, 0 if enabled.

So the SIM and RIM instructions are typically used to either output to or input from 8085serial interface, or enable/disable/read the interrupt masks for interrupts 7.5, 6.5, 5.5, but usually not at the same time.

3) EI - ENABLE INTERRUPT INSTRUCTION

This is a single byte instruction. On execution interrupt enable and all interrupts are enabled. No flags are affected.

Opcode	Operand	Bytes	Machine Cycles	T–states
EI	––	1	1	4

4) GENERAL MACHINE CONTROL OPERATIONS.

These are for CPU halting, no operation etc. Following are the instructions.

1) PCHL - LOAD PROGRAM COUNTER WITH HL REGISTER PAIR CONTENTS.

This is a single byte instruction. It copies contents of HL pair into PC.

The result is equivalent is to 1-byte unconditional jump with address stored in HL pair. No flags are affected.

Opcode	Operand	Bytes	Machine Cycles	T–states
PCHL	––	1	1	4

2) NOP-NO OPERATION

This is a single byte instruction. On execution no operation is performed only instruction is fetched and decoded. No flags are affected. It can be create time delays using loops.

3) HLT -HALT AND ENTER WAIT STATE

This is a single byte instruction. After completing its execution CPU goes to a halt state halting further execution. Wait states are inserted in every clock cycle. During Halt CPU maintains register contents. But tri-states address and data lines. An interrupt or reset is necessary to exit from Halt state.

This completes machine control group instructions.

Exercise

Select the correct alternative and rewrite the following.

1. ……. instruction belongs to data transfer group of instruction set of 8085.

(i) LHLD (ii) CMA (iii) JMP (iv) POP

1. (i) LHLD

2. ..... flag is affected by the instruction RRC of 8085.

(i) zero (ii) parity (iii) carry (iv) all

2. (iii) carry

3. Which of the following instruction does not affect any flag.....

(i) ADD (ii) RAR (iii) STC (iv) PCHL

3. (iv) PCHL

4. Instruction STAX belongs to addressing mode.

(i) Direct (ii) Register

(iii) Register indirect (iv) Immediate

4. (iii) Register indirect

5. In 8085.....instruction affects flag register.

(i) MOV B, A (ii) CMA (iii) MVI A, data (iv) CPI data

5. (iv) CPI data

6. Instruction PCHL belongs to.....group

(i) Arithmetic operation (ii) Logical operation

(iii) Data transfer (iv) Branching operation

6. (iv) Branching operation

7. LXI H, addr is byte instruction.
(i) 1 (ii) 2 (iii) 3 (iv) 4
7. (iii) 3

8. The contents of H-L pair are 29 FF. After execution of the instruction INX H, the contents will be.....
(i) 2A00 (ii) 3000 (iii) 2910 (iv) 2900
8. (i) 2A00

9. DAD instruction only affects flag.
(i) Parity (ii) Auxiliary carry (iii) Carry (iv) All of these
9. (iii) Carry

10. After execution of ANA instruction, Cy flag is.....and Ac flag is.....
(i) reset, set (ii) set, reset (iii) set, set (iv) reset, reset
10. (i) reset, set

11. The instruction that can affect the stack pointer is.....
(i) SHLD (ii) XCHG (iii) LXI (iv) LDAX
11. (iii) LXI

12. ……. is three byte instruction.
(i) RAR (ii) MOV A, D (iii) SBI 80H (iv) LXI H, 2050H
12. (iv) LXI H, 2050H •

13. The invalid register pair for 8085 microprocessor is .....
(i) BC (ii) HL (iii) SP (iv) DE
13. (iii) SP

14. The instruction XRA M comes under the category of group.
(i) Data transfer (ii) Branch control (iii) Logical (iv) Arithmetic
14. (iii) Logical

15. The instruction PCHL belongs to addressing mode.
(i) register indirect (ii) direct (iii) register (iv) implicit
15. (iii) register

16. There is no branch instruction based upon the ..... flag.
(i) CY (ii) S (iii) AC (iv) P
16. (iii) AC

17. The instruction MOV B, A of 8085 microprocessor is an example of addressing mode.
(i) Direct (ii) Implicit
(iii) Register indirect (iv) Register
17. (iv) Register

18. In 8085 microprocessor, flag register is not affected after the execution of ……. instruction,

(i) INR r (ii) DCR r (iii) ADD r (iv) INX rp

18. (iv) INX rp

19. .....of following instruction belongs to register indirect addressing mode.

(i) LXI H, 1050 (ii) MVI A, 05 (iii) CMP B (iv) MOV C, M

19. (iv) MOV C, M

20. The full form of instruction DAA is ……

(i) Double Add Accumulator (ii) Decimal Add Accumulator

(iii) Double Adjust Accumulator (iv) Decimal Adjust Accumulator

20. (iv) Decimal Adjust Accumulator

21. The instruction which does not affect only carry flag is .....

(i) DAD (ii) XRA (iii) CMP (iv) INR

21. (iv) INR

22. .....instruction rotates the contents of ACC left through carry by 1 bit.

(i) RLC (ii) RRC (iii) RAR (iv) RAL

22. (iv) RAL

23. ACC contents remain unchanged on execution of instruction .....

(i) LDAX rp (ii) MOV A, M (iii) CMA (iv) CMP B

23. (iv) CMP B

24. The instruction MOV B, A of 8085 microprocessor is an example of addressing mode.

(i) Direct (ii) Implicit

(iii) Register indirect (iv) Register

24. (iv) Register

25. In 8085 microprocessor, flag register is not affected after the execution of instruction.

(i) INRr (ii) DCRr (iii) ADD r (iv) INXrp

25. (iv) INX rp

26. ……of following instruction belongs to register indirect addressing mode.

(i) LXI H, 1050 (ii) MVI A, 05 (iii) CMP B (iv) MOV C, M

26. (iv) MOV C, M

27. The full form of instruction DAA is ……

(i) Double Add Accumulator (ii) Decimal Add Accumulator

(iii) Double Adjust Accumulator (iv) Decimal Adjust Accumulator

27. (iv) Decimal Adjust Accumulator

28. The instruction which does not affect, only carry flag is ……..

(i) DAD (ii) XRA (iii) CMP (iv) INR

28. (iv) INR

29. ……..instruction rotates the contents of ACC left through carry by 1 bit.

(i) RLC (ii) RRC (iii) RAR (iv) RAL

29. (iv) RAL

30. AC contents remain unchanged on execution of instruction ……...

(i) LDAX rp (ii) MOV A, M (iii) CMA (iv) CMP B

30. (iv) CMP B

31. The instruction which affects only carry flag is …….

(i) OR (ii) XRI (iii) ADI (iv) DAD

31. (iv) DAD

32. ……instruction uses flags.

(i) Data Transfer (ii) Arithmetical

(iii) Conditional jump (iv) Logical

32. (iii) Conditional Jump

33. The instruction that can affect stack pointer is…….

(i) SHLD (ii) XCHG (iii) LXI (iv) LDAX

33. (iii) LXI

34. The instruction …….. will affect the zero flag without changing the contents of the accumulator.

(i) MVI A, 00 (ii) SUB A (iii) XRA A (iv) CMP A

34. (iv) CMP A

35. XCHG instruction exchanges 16-bit data between ……..

(i) DE and HL Register Pair (ii) BC and HL Register Pair

(iii) BC and DE Register Pair (iv) All of the above Register Pairs

35. (i) DE and HL Register Pair

36. In case of 8085 instructions, STC is an example of ……. addressing mode.

(i) Direct (ii) Register (iii) Implied (iv) Immediate

36. (iii) Implied

37. Addressing Mode of ADD M is ……..

(i) Direct (ii) Register Indirect

(iii) Implied (iv) Immediate

37. (ii) Register Indirect

38. …… of the following instructions is branching instruction.
(i) ADD r (ii) JMP addr (iii) CMP M (iv) CMA
38. (ii) JMP addr

39. In case of 8085 instruction set, CMA is an example of ……. instruction.
(i) Arithmetic (ii) Branching (iii) Logical (iv) Data Transfer
39. (iii) Logical

40. During PUSH instruction of 8085, the stack pointer_
(i) Increment by 1 (ii) Increment by 2
(iii) Decrement by 1 (iv) Decrement by 2
40. (iv) Decrement by 2

41. PSW is a combination of …….registers.
(i) M and F (ii) H and F (iii) L and F (iv) A and F
41. (iv) A and F

42. The instruction CMA is……Byte function.
(i) 1 byte (ii) 2 byte (iii) 3 byte (iv) 4 byte
42. (i) 1 byte

43. ……instruction is Logical Instruction.
(i) ADD r (ii) MVI r, data (iii) ANI, data (iv) LXl rp, data
43. (iii) ANI, data

44. The-instruction JNZ of 8085 microprocessor is …… type of instruction.
(i) Branching (ii) Conditional Branching
(iii) Arithmetic (iv) Data Transfer
44. (ii) Conditional Branching

45. …… Flag is always reset in ANA instruction.
(i) Carry (ii) Parity (iii) Sign (iv) Zero
45. (i) Carry

46. The flag bit that gets affected on execution of RCC instruction in 8085 Processor is …….
(i) Zero (ii) Parity (iii) Carry (iv) All
46. (iii) Carry

47. ……flag is affected in CMA Instruction.
(i) All (ii) No (iii) Carry (iv) Zero
47. (ii) No

48. LXI rp, $Data_{16}$ is ……. Byte instruction.
(i) TWO (ii) ONE (iii) THREE (iv) FOUR
48. (iii) THREE

49. …….. instruction would not affect zero flag.
(i) XRA A (ii) SUB A (iii) CMP A (iv) MVI A, 00H
49. (iv) MVI A, 00H

50. After the execution of POP rp instruction, SP gets …….
(i) Incremented by one (ii) Decremented by one
(iii) Incremented by two (iv) Decremented by two
50. (iii)

51. ……. Instruction does not affect the Flag.
(i) RAR (ii) CMP C (iii) XRA (iv) MOV A, B
51. (iv) MOV A, B

52. ……..instruction is used for 16-bit addition.
(i) ADD (ii) ADI (iii) ADC (iv) DAD
52. (iv)

53. In MOV A, M instruction …… is used to point the memory location.
(i) HL (ii) PC (iii) SP (iv) PSW
53. (i) HL

54. ANA, r instruction comes under …… group.
(i) Arithmetic (ii) Logical (iii) Branch (iv) Data Transfer
54. (ii)

55. …… is three-byte instruction of 8085.
(i) CMA (ii) ADI (iii) XCHG (iv) LDA
55. (iv) LDA

56. The instruction PCHL belongs to …….. group.
(i) Data transfer (ii) Logical (iii) Arithmetic (iv) Branching
56. (iv) Branching

57. …….. instruction is a Arithmetic group of instruction.
(i) MOV reg, reg (ii) RRC (iii) NOP (iv) ADD reg.
57. (iv) ADD reg.

58. The first byte of an 8085 instruction always contains …….
(i) Opcode (ii) Data (iii) Address (iv) None of these
58. (i) Opcode

59. The PUSH PSW instruction of 8085 shall …… the stack pointer.
(i) Increment by two bytes (ii) Decrement by two bytes
(iii) Un affect (iv) None of these
59. (ii) Decrement by two bytes

60. The length of instruction MVI reg. data is ……
(i) 1 Byte (ii) 2 Byte (iii) 3 Byte (iv) 4 Byte
60. (ii) 2 Byte

❑ ❑ ❑ ❑ ❑ ❑