Registers

The 8085 has six general-purpose registers to store 8-bit data; these are identified as B, C, D, E, H, and L as shown in the Fig. 5.2.1. They can be combined as register pairs - BC, DE, and HL - to perform some 16-bit operations. The programmer can use these registers to store or copy data into the registers by using data copy instructions.

Accumulator

The accumulator is an 8-bit register that is a part of arithmetic / logic unit (ALU). This register is used to store 8-bit data and to perform arithmetic and logical operations. The result of an operation is stored in the accumulator. The accumulator is also identified as register A.

Flags

The ALU includes five flip-flops, which are set or reset after an operation according to data conditions of the result in the accumulator and other registers. They are called Zero(Z), Carry (CY) , Sign (S), Parity (P), and Auxiliary Carry (AC) flags. The most commonly used flags are Zero, Carry, and Sign. The microprocessor uses these flags to test data conditions.

For example, after an addition of two numbers, if the sum in the accumulator is larger than eight bits, the flip-flop used to indicate a carry called the Carry flag (CY) is set to one. When an arithmetic operation results in zero, the flip-flop called the Zero (Z) flag is set to one. Fig. 5.2.1 shows an 8-bit register, called the flag register, adjacent to the accumulator. However, it is not used as a register; five bit positions out of eight are used to store the outputs of the five flip-flops. The flags are stored in the 8-bit register so that the programmer can examine these flags (data conditions) by accessing the register through an instruction. These flags have critical importance in the decision-making process of the microprocessor. The conditions (set or reset) of the flags are tested through the software instructions. For example, the instruction JC (Jump on Carry) is implemented to change the sequence of a program when CY flag is set. The thorough understanding of flag is essential in writing assembly language programs.

Program Counter (PC)

This 16-bit register deals with sequencing the execution of instructions. This register is a memory pointer. 8085 has 16-bit addresses, and that is why this is a 16-bit register. The microprocessor uses this register to sequence the execution of the instructions. The function of the program counter is to point to the memory address from which the next byte is to be fetched. When a byte (machine code) is being fetched, the program counter is incremented by one to point to the next memory location.

Stack Pointer (SP)

- The stack pointer is also a 16-bit register used as a memory pointer. It points to a memory location in R/W memory, called the stack. The beginning of the stack is defined by loading 16-bit address in the stack pointer.

5.3 Instruction Formats

Q. What do you mean by instruction format in 8085 MPU? Describe the same by giving suitable examples.

- An **instruction** is a binary pattern designed inside a microprocessor to perform a specific function. The entire group of instructions, called the **instruction set**, determines what functions the microprocessor can perform.
- Each instruction basically consists of two parts. The first part is called as **opcode** and the second is called as ***operand***.
- The opcode part of an instruction specifies the operation to be performed. The operands either provides the data or specifies data. The operand can be specified in a number of ways. It includes :
  (i) 8 bit/16 bit internal general purpose register.
  (ii) A memory location.
  (iii)8 bit port address/16 bit memory address.
  (iv)Implicit operand : The operand is not specified, instead it is assumed in register.

Depending on the number of bytes or bit patterns required to specify an operation and the data, the instructions are of 3 types :

- One byte instructions • Two byte instructions. • Three byte instructions.

5.3.1 1 Byte Instructions

- A 1 byte instruction includes the opcode and the operand in the 8 bit only i.e. one byte.
- The one byte instructions specify the operation to be performed and who is going to perform it. These, instructions, require one (single) memory location.

Format :

Opcode

Examples

1) **MOV B,C**
   - This instruction moves the contents of register C in register B.
   - This instruction has an opcode 41 H and is a one byte instruction . This instruction copies the contents of register C in B register .

5.3.2 2 Bytes Instructions

Q. Explain two byte and three byte instructions.

- The 2 bytes instruction uses first byte to specify the operation i.e. opcode and second byte to specify the operand.
- These instructions require two successive memory locations in the memory.
- First byte stored is opcode and second byte stored is data i.e. operand or address.

Format

Opcode	Operand

Example

MVI B,57H
   - This instruction moves the data 57H in register B.
   - The first byte of this instruction is the opcode for instruction MVI B , and the second byte is the data that is to be moved to register B .

5.3.3 3 Byte Instructions

- The 3 bytes instruction uses first byte to specify the operation i.e. opcode, second and third bytes are used to specify the operand. Generally these instructions are used to specify 16 bit data or memory address with the instruction.
- First byte stored is opcode, second byte stored is lower order 8 bits of 16 bits operand or address and third byte stored is higher order 8 bits of 16 bits operand or address.

Format

Opcode	Operand	Operand

Example

STA 5600 H

- This instruction moves the contents of accumulator to the memory location 5600H.
- The first byte of this instruction is the opcode for instruction STA 5600 H , and the second and third bytes are the address 5600 H .

5.3.4 Opcode Formats

- The microprocessor 8085 is an 8 bit microprocessor and has 8 bit opcodes. Each instruction has a unique opcode.
- The opcode contains information regarding the operation , type of operation to be performed , registers to be used , flags . The opcode is fixed for each instruction .

Notations used in object code or OPCODE are :

Table 5.3.1(a)

Notations	Meaning
ddd	Destination register(s)
sss	Source registers, ddd = sss = 111 = A register 000 = B register 001 = C register 010 = D register 011 = E register 100 = H register 101 = L register
nnn	Restart number 000 to 111
yy	An 8 bit binary data
yyyy	A 16 bit binary data unit
x	Register pair 0 = BC 1 = DE
xx	Register pair 00 = BC 01 = DE 10 = HL 11 = SP (if PUSH/POP) PSW
PPQQ	A 16 bit memory address

Table 5.3.1(b)

Information operations			
			Operation
I_o	I_o	I_o	
0	0	0	Read/set interrupt mask
0	0	1	Immediate operation R_p
0	1	0	Load/store
0	1	1	Increment/Decrement R_p
1	0	0	Increment single register
1	0	1	Decrement single register
1	1	0	Immediate operation on single register.
1	1	1	Register shifting / Miscellaneous.

Table 5.3.1(c)

Arithmetic logic unit operations			
A_L	A_L	A_L	Operati
0	0	0	ADD
0	0	1	ADD with c
0	1	0	SUB
0	1	1	SUB with bo
1	0	0	Logical AND
1	0	1	X-OR
1	1	0	Logical OR
1	1	1	Compare

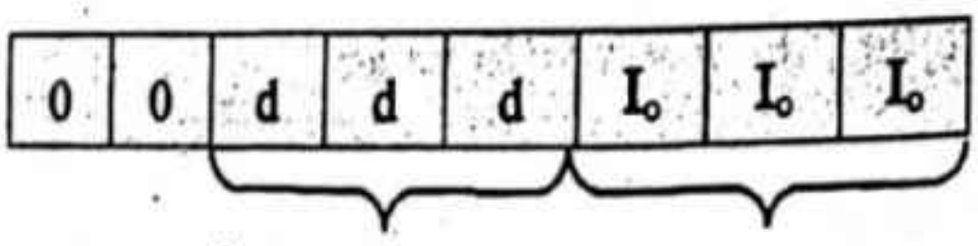
Table 5.3.1(d)

Branch condition			
			Operation
C_B	C_B	C_B	
0	0	0	JNZ
0	0	1	JZ
0	1	0	JNC
0	1	1	JC
1	0	0	JPO
1	0	1	JPE
1	1	0	JP
1	1	1	JM

Table 5.3.1(e)

Branch operation			
			Operation
B_o	B_o	B_o	
0	0	0	Conditional return
0	0	1	Simple return / Miscellaneou
0	1	0	Conditional jump
0	1	1	Unconditional jump / Miscellaneous
1	0	0	Conditional CALL
1	0	1	Simple CALL / Miscellaneou
1	1	0	Special A / L operations
1	1	1	Special unconditional jump

For all the data transfer instructions except MOV instruction format of the is,



5.3.5 Data Formats

- The microprocessor 8085 is an 8 bit microprocessor. It processes only numbers.
- However the real world operates in decimal numbers, characters and alph Hence, we need to code the binary numbers.
- The commonly used codes and data formats for 8 bit microprocessor system are

i) **ASCII code :** It is a 7 bit alphanumeric code representing decimal numbers, english alphabets and nonprintable characters like carriage return. Extended ASCII is an 8 bit code.

ii) **BCD code :** (Binary coded decimal). It is used for representing decimal numbers from 0 to 9. To do this only four bits are required ranging from 0000 to 1001. The remaining numbers are considered as invalid. Two BCD digits can be stored in register of microprocessor 8085.

iii) **Signed integer :** A signed integer is a positive or a negative number. The most significant digit i.e. D_7 is used to present the sign of a number. If this digit is 1 ; it indicates a negative number whereas if digit is 0, it indicates a positive number. The remaining bits represent the magnitude of the number. Hence, the largest positive number at a time can be 0111 1111 (7FH). While the remaining numbers from 80 H to FFH are considered to be negative numbers. The negative numbers in microprocessor 8085 are represented in the 2's complement form.

iv) **Unsigned integers :** An integer without a sign can be represented by all the 8 bits in the microprocessor register. Hence, the largest number that can be processed at a time is FFH.

Ex. 5.3.1 : Define opcode and operand. Write machine code for MOV H, A If opcode – 01_2, register code H = 100_2 and A = 111_2

Soln. :

Machine code :

Address	Memory	
2000 H	01	Opcode of MOV H, A
2001 H		Next instruction.

Ex. 5.3.2 : Define opcode, operand and instruction format. Derive the opcode for instruction MOV A, H if code for MOV = $(01)_2$, register code for H = $(100)_2$ and register code for A = $(111)_2$

Soln. :

The format of opcode for MOV instructions is,

0	1	d	d	d	s	s	s

Hence, opcode for the instruction MOV A,H is,

0	1	1	1	1	1	0	0

= 7CH

5.4 Addressing Modes

Q. Mention the various addressing modes in 8085 microprocessor with a corresponding example.

- The different methods (modes) to select (address) the operands are called the **addressing modes.** For 8085, they are :

- Immediate addressing.
- Register addressing.
- Direct addressing.
- Indirect addressing.
- Implied addressing

5.4.1 Immediate Addressing Mode

- In immediate addressing mode the data (8 / 16 bit) is specified in the instruction itself.
- The immediate addressing instructions are either 2 byte or 3 byte long. In 2 byte instruction first byte is OPCODE and second byte is the 8 bit data. In 3 byte instructions first byte is OPCODE, second and third bytes are the 16 bit data.
- The instructions containing the letter "I" indicate immediate addressing mode.
- Fig. 5.4.1 shows the location of operand.

Fig. 5.4.1 : Immediate addressing mode instruction format

Examples :

(i) MVI A, A0 H :
This instruction transfers immediate data (A0 H) to A register.

(ii) LXI H, C200 H :
This instruction transfers 16 bit immediate data C200 to HL register pair.
Lower order data (00 H) to L register and high order data (C2 H) to H register.

5.4.2 Register Addressing Mode

- In register addressing mode the source and destination operands are general purpose registers.
- The register addressing instructions are generally of 1 byte
  i.e. OPCODE only. The OPCODE specifies the operation and registers to be used to perform the operation.
- Fig. 5.4.2 shows the location of operand.

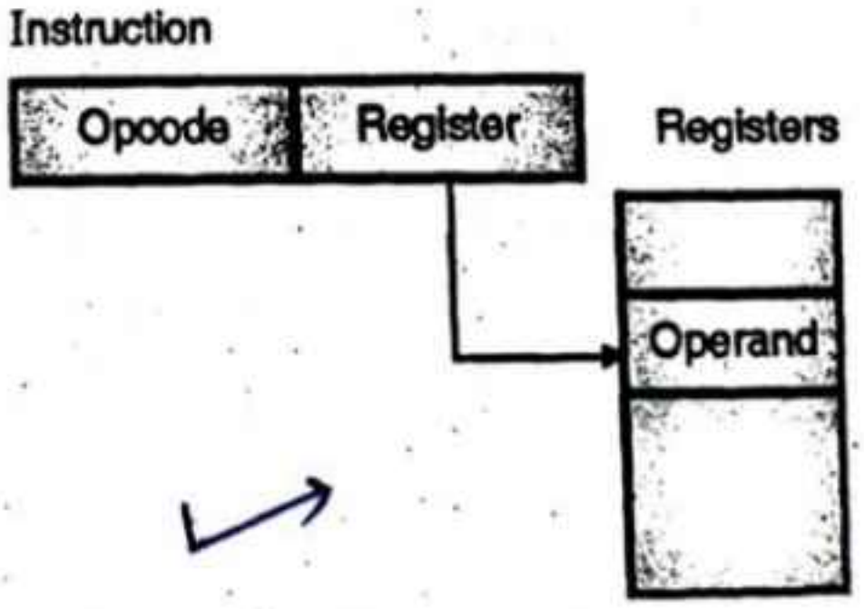


Fig. 5.4.2 : Register addressing mode instruction format

Examples

(i) MOV D, B : This instruction copies the contents of register B to the D register. The source and destination operands are both registers.

(ii) ADD B : This instruction adds the contents of B register and A register. The data is present in both B and A registers. The result is stored in the accumulator .

(iii) PCHL : This instruction will transfer the contents of register pair HL to the PC.

5.4.3 Direct Addressing Mode

- In direct addressing mode the 16 bit address of the operand is given within the instruction itself.

- The instructions in the direct addressing mode are 3 byte instructions. First byte is a OPCODE, second is lower order address byte and third is higher order address byte.
- For I/O instruction that use direct addressing mode are 2 byte as the address of I/O is one byte.
- Fig. 5.4.3 shows the location of operand.

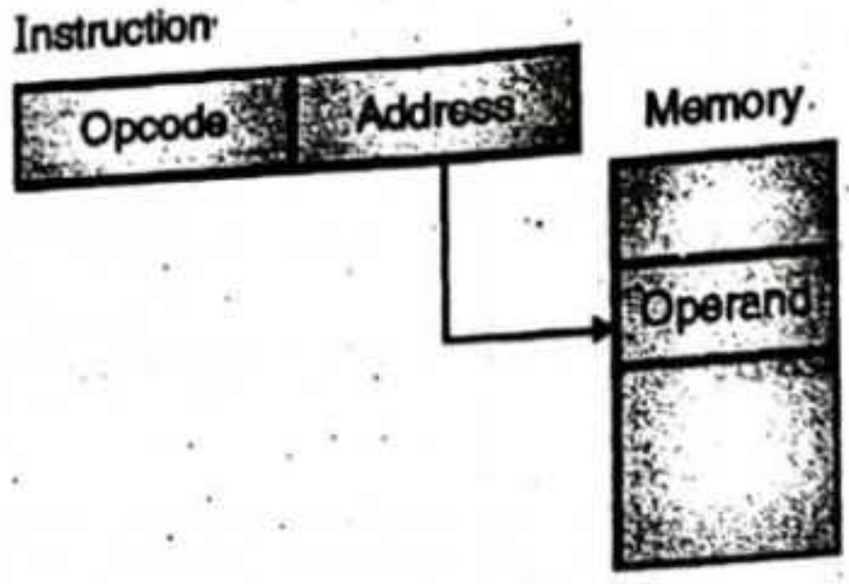


Fig. 5.4.3 : Direct addressing mode instruction format

Examples

(i) LDA C200 H : Load accumulator directly from memory location. In this instruction the contents of C200 memory location are transferred to accumulator.

(ii) STA C200 H : Store accumulator directly to memory location. In this instruction the contents of accumulator are stored at memory location C200 H.

5.4.4 Indirect Addressing Mode

- In indirect addressing mode the instructions reference the memory through a register pair. i.e. the memory address where the operand is located is specified by the contents of a register pair.
- Fig. 5.4.4 shows the location of operand.

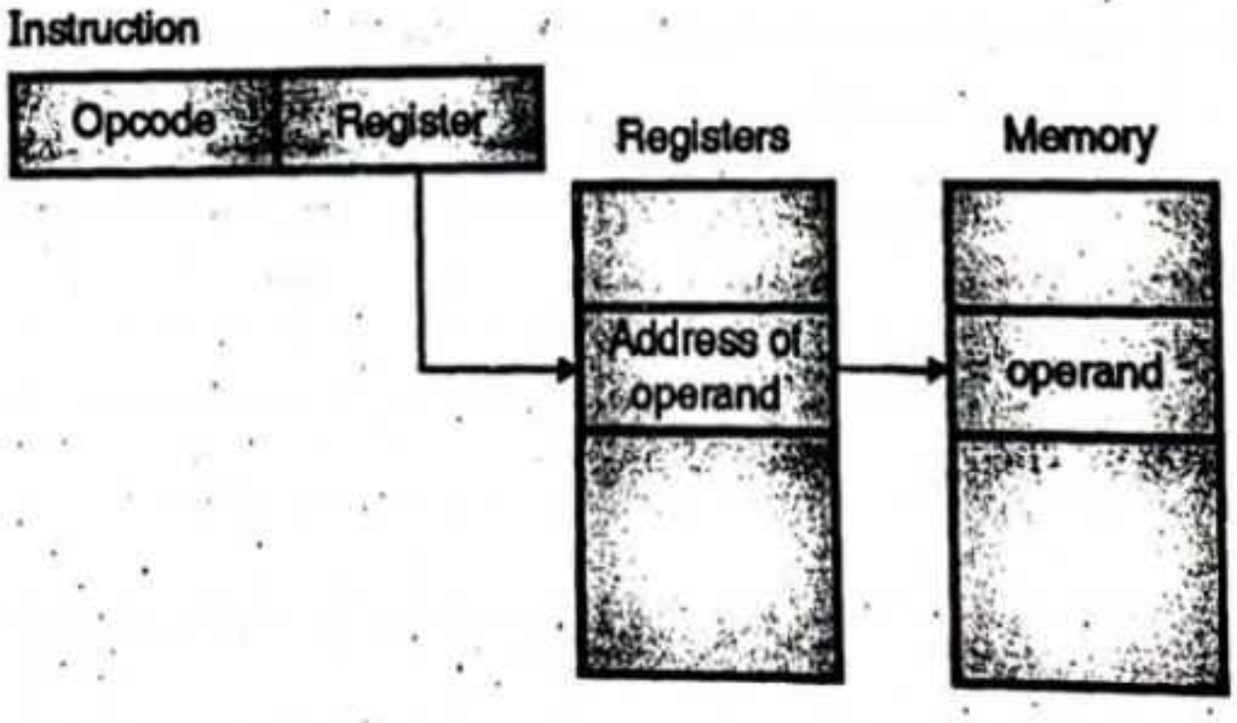


Fig. 5.4.4 : Indirect addressing mode instruction format

Examples

(i) MOV A,M : In this case M is a memory pointer specifying HL register pair where the address is stored. The contents of HL pair are used as address and the contents of that memory location are transferred to accumulator.

(ii) LDAX B : In this case BC register pair is used as address and the contents of the memory location specified by BC register pair are copied to accumulator.

Register of M.L contents
of that M.L are

5.4.5 Implied Addressing or Implicit Addressing Mode

- The implied mode of addressing does not require any operand.
- The data is specified within the OPCODE itself. Generally, the implied addressing mode instruction is a 1 byte instruction.
- The data is supposed to be present generally in accumulator.

Example

(i) RAL : Rotate accumulator left, it operates on the data in accumulator only. So whenever RAL is used it is implied that the data to be operated on is available in the accumulator only.

(ii) CMC : Complement carry flag.

5.5 Introduction to Programming : Writing, Assembling and Executing a Program

Q. List the major steps in developing an assembly language program.

The 8085 is capable of performing a few operations at very high speed. 808 instruction set indicates that :

1. The ALU in combination with various registers can be controlled by binar operational codes to perform arithmetic operations.
2. Microprocessor can transfer data inside the CPU.
3. Microprocessor can perform logical and arithmetic manipulations.
4. Microprocessor can make decisions based on the status of the result and
5. Microprocessor can move data into and out of the microprocessor.

- The microprocessor cannot, by itself, perform simple calculations or data transfer. I must be told in clear terms, in complete detail, just what to do and in what orde The physical components, connection and logic circuits that take part in decodin and execution of operation codes constitute the computer hardware.
- The lists of specific instructions, selected from those allowed by the microprocesso manufacturer and organized to control operations constitute computer software.

We will study the programming techniques and programming in the followin chapters. The programming is divided into two parts. Simple programs and after that concept of looping is introduced for programs.

Programming Steps

In programming technique five different points have been given, we follow those :

Step 1 :	Define the problem to be solved
Step 2 :	Solution plan
Step 3 :	Flowchart
Step 4 :	Program
Step 5 :	Check the result

C BC
A DD
199

1 1 1 1 1 1 1 1
1 0 1 1 1 1 0 0
1 1 0 1 1 1 0 1
1 0 0 1 1 0 0 1

The following abbreviations are used in the description of the instruction set.

Flags

Reg.	= 8080A/8085 Register	S	= Sign
Mem.	= Memory Location	Z	= Zero
R	= Register	AC	= Auxiliary Carry
Rs	= Register Source	P	= Parity
Rd	= Register Destination	CY	= Carry
M	= Memory		
( )	= Contents of		
XX	= Random Information		

ACI: Add Immediate to Accumulator with Carry

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
ACI	8-bit data	2	2	7	CE

Description The 8-bit data (operand) and the Carry flag are added to the contents of the accumulator, and the result is stored in the accumulator.

Flags All flags are modified to reflect the result of the addition.

Example Assuming the accumulator contains 26H and the previous operation has set the Carry flag, add byte 57H to the accumulator.

Instruction: ACI 57H **Hex Code:** CE 57

even P=1
odd P=0

Addition:

(A): 26H = 0 0 1 0 0 1 1 0
(Data): 57H = 0 1 0 1 0 1 1 1
CY 1 = 1
7EH = 0 1 1 1 1 1 1 0
Flags: S = 0 Z = 0 AC = 0
P = 1 CY = 0

Comments:

1. After addition the previous Carry flag is cleared.
2. This instruction is commonly used in 16-bit addition. This instruction should not be used to account for a carry generated by 8-bit numbers.

Z = 0
P = 1 even
CY = 1
AC = 1
S = 1 MSB of A

16

ADC: Add Register to Accumulator with Carry

Opcode	Operand	Bytes	M-Cycles	T-States
ADC	Reg.	1	1	4
	Mem.	1	2	7

Hex Codes

Reg.	Hex
B	88
C	89
D	8A
E	8B
H	8C
L	8D
M	8E
A	8F

Description The contents of the operand (register or memory) and the Carry flag are added to the contents of the accumulator and the result is placed in the accumulator. The contents of the operand are not altered; however, the previous Carry flag is reset.

Flags All flags are modified to reflect the result of the addition.

Example Assume register pair BC contains 2498H and register pair DE contains 54A1H. Add these 16-bit numbers and save the result in BC registers.

The steps in adding 16-bit numbers are as follows:

1. Add the contents of registers C and E by placing the contents of one register in the accumulator. This addition generates a Carry. Use instruction ADD (explained on the next page) and save the low-order 8-bits in register C.

```
      98H =      1 0 0 1  1 0 0 0
      A1H =      1 0 1 0  0 0 0 1
    ------------------------------
    1 39H = 1    0 0 1 1  1 0 0 1     Store in register C
    CY       CY
```

2. Add the contents of registers B and D by placing the contents of one register in the accumulator. Use instruction ADC.

   The result will be as follows.

```
      24H = 0 0 1 0  0 1 0 0
      54H = 0 1 0 1  0 1 0 0
      1   =                1     (Carry from the previous addition)
    ------------------------------
      79H = 0 1 1 1  1 0 0 1     Store in register B
```

Comments: This instruction is generally used in 16-bit addition. For example, to add the contents of BC registers to the contents of DE registers this instruction is used to account for the carry generated by low-order bytes.

ADD: Add Register to Accumulator

Opcode	Operand	Bytes	M-Cycles	T-States
ADD	Reg.	1	1	4
	Mem.	1	2	7

Hex Codes

Reg.	Hex
B	80
C	81
D	82
E	83
H	84
L	85
M	86
A	87

Description The contents of the operand (register or memory) are added to the contents of the accumulator and the result is stored in the accumulator. If the operand is a memory location, that is indicated by the 16-bit address in the HL register.

Flags All flags are modified to reflect the result of the addition.

Example Register B has 51H and the accumulator has 47H. Add the contents of register B to the contents of the accumulator.

Instruction: ADD B **Hex Code:** 80

Register contents before instruction

A	47	X	F
B	51	X	C

Addition

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

Register contents after instruction

		S Z AC P CY	
A	98	1 0 0 0 0	F
B	51	X	C

Flags: S = 1, Z = 0, AC = 0
P = 0, CY = 0

Example Memory location 2050H has data byte A2H and the accumulator has 76H. Add the contents of the memory location to the contents of the accumulator.

Instruction: ADD M **Hex Code:** 86

Before this instruction is executed, registers HL should be loaded with data 2050H.

Register contents
before instruction

A	76	X	F
B	X	X	C
D	X	X	E
H	20	50	L

2050 | A2 |

Addition:

(A) 76H = 0 1 1 1 0 1 1 0
(2050H)Mem A2H = 1 0 1 0 0 0 1 0
1/18H = 1/0 0 0 1 1 0 0 0
CY CY

Register contents
after instruction

	S Z AC P CY	
A	18	0 0 0 1 1 F
B	X	X C
D	X	X E
H	20	50 L

Flags: S = 0, Z = 0, AC = 0,
P = 1, CY = 1

ADI: Add Immediate to Accumulator

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Codes
ADI	8-bit data	2	2	7	C6

Description The 8-bit data (operand) are added to the contents of the accumulator, and the result is placed in the accumulator.

Flags All flags are modified to reflect the result of the addition.

Example The accumulator contains 4AH. Add the data byte 59H to the contents of the accumulator.

Instruction: ADI 59H Hex Code: C6 59

Addition:

(A) : 4AH = 0 1 0 0 1 0 1 0
+
(Data) : 59H = 0 1 0 1 1 0 0 1
A3H = 1 0 1 0 0 0 1 1

Flags: S = 1, Z = 0, AC = 1
P = 1, CY = 0

ANA: Logical AND with Accumulator

Opcode	Operand	Bytes	M-Cycles	T-States
ANA	Reg.	1	1	4
	Mem.	1	2	7

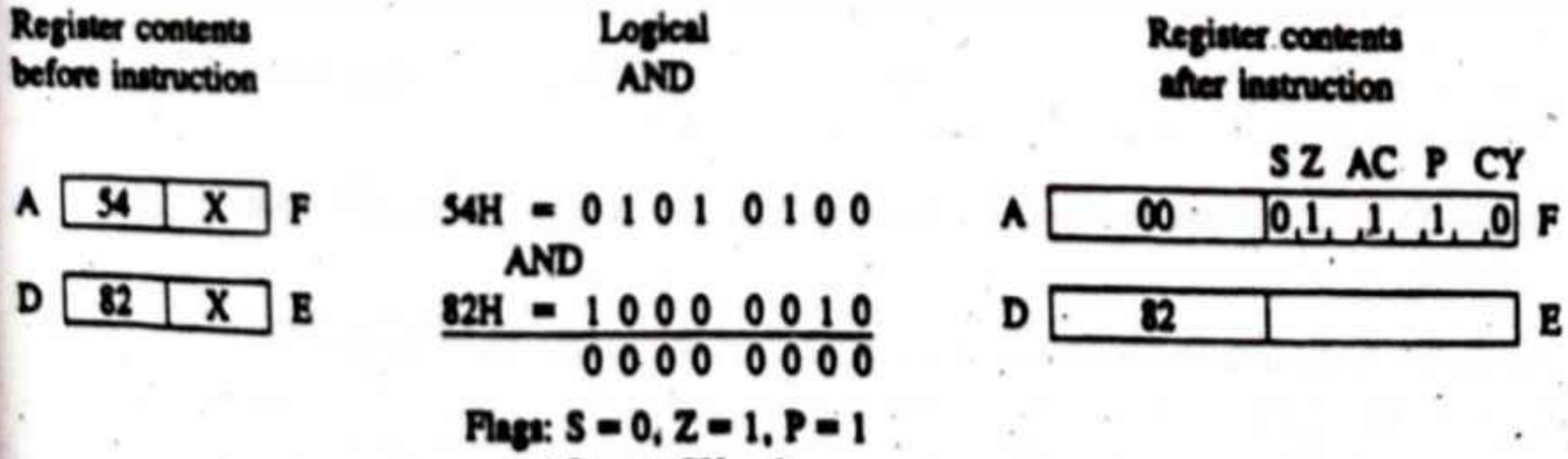
Reg.	Hex
B	A0
C	A1
D	A2
E	A3
H	A4
L	A5
M	A6
A	A7

Description The contents of the accumulator are logically ANDed with the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers.

Flags S, Z, P are modified to reflect the result of the operation. CY is reset. In 8085, AC is set, and in 8080A AC is the result of ORing bits D_3 of the operands.

Example The contents of the accumulator and the register D are 54H and 82H, respectively. Logically AND the contents of register D with the contents of the accumulator. Show the flags and the contents of each register after ANDing.

Instruction: ANA D Hex Code: A2



ANI: AND Immediate with Accumulator

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
ANI	8-bit data	2	2	7	E6

Description The contents of the accumulator are logically ANDed with the 8-bit data (operand) and the results are placed in the accumulator.

Flags S, Z, P are modified to reflect the results of the operation. CY is reset. In 8085, AC is set.

Example AND data byte 97H with the contents of the accumulator, which contains A3H.

Instruction: ANI 97H **Hex Code:** E6 97

Logical AND:

```
(A)    : A3H = 1 0 1 0  0 0 1 1
               AND
(Data) : 97H = 1 0 0 1  0 1 1 1
               ---------------
               1 0 0 0  0 0 1 1
```

	S	Z	AC	P	CY	
A	83	1	0	1	0	0 F

CALL: Unconditional Subroutine Call

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
CALL	16-bit address	3	5	18	CD

Description The program sequence is transferred to the address specified by the operand. Before the transfer, the address of the next instruction to CALL (the contents of the program counter) is pushed on the stack. The sequence of events is described in the example below.

Flags No flags are affected.

Example Write CALL instruction at memory location 2010H to call a subroutine located at 2050H. Explain the sequence of events when the stack pointer is at location 2099H.

Memory Address	Hex Code	Mnemonics
2010	CD	CALL 2050H
2011	50	
2012	20	

Note: See the difference between writing a 16-bit address as mnemonics and code. In the code, the low-order byte (50) is entered first, then the high-order byte (20) is entered. However, in mnemonics the address is shown in the proper sequence. If an assembler is used to obtain the codes, it will automatically reverse the sequence of the mnemonics.

Execution of CALL: The address in the program counter (2013H) is placed on the stack as follows.

2097	13
2098	20

SP → 2099

Stack pointer is decremented to 2098H
MSB is stored
Stack pointer is again decremented
LSB is stored
Call address (2050H) is temporarily stored in internal WZ registers and placed on the bus for the fetch cycle

Comments: The CALL instruction should be accompanied by one of the return (RET or conditional return) instructions in the subroutine.

Conditional Call to Subroutine **Operand—16-Bit Address**

Op Code	Description	Flag Status	Hex Code	M-Cycles T-States
CC	Call on Carry	CY = 1	DC	2/9 (if condition is not true)
CNC	Call with No Carry	CY = 0	D4	5/18 (if condition is true)
CP	Call on positive	S = 0	F4	*Note:* If condition is not true it continues
CM	Call on minus	S = 1	FC	the sequence, and thus requires
CPE	Call on Parity Even	P = 1	EC	fewer T-states.
CPO	Call on Parity Odd	P = 0	E4	If condition is true it calls the
CZ	Call on Zero	Z = 1	CC	subroutine, thus requires more
CNZ	Call on No Zero	Z = 0	C4	T-states.

Flags No flags are affected.

CMA: Complement Accumulator

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
CMA	None	1	1	4	2F

Description The contents of the accumulator are complemented.

Flags No flags are affected.

Example Complement the accumulator, which has data byte 89H.

Instruction: CMA Hex Code: 2F

Before instruction

A | 1 0 0 0 1 0 0 1 | = 89H

After instruction

A | 0 1 1 1 0 1 1 0 | = 76H

CMC: Complement Carry

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
CMC	None	1	1	4	3F

Description The Carry flag is complemented.

Flags The Carry flag is modified, no other flags are affected.

CMP: Compare with Accumulator

Opcode	Operand	Bytes	M-Cycles	T-States
CMP	Reg.	1	1	4
	Mem.	1	2	7

Hex Codes

Reg.	Hex
B	B8
C	B9
D	BA
E	BB
H	BC
L	BD
M	BE
A	BF

Description The contents of the operand (register or memory) are compared with the contents of the accumulator. Both contents are preserved and the comparison is shown by setting the flags as follows:

- If (A) < (Reg/Mem): Carry flag is set and Zero flag is reset.
- If (A) = (Reg/Mem): Zero flag is set and Carry flag is reset.
- If (A) > (Reg/Mem): Carry and Zero flags are reset.

The comparison of two bytes is performed by subtracting the contents of the operand from the contents of the accumulator; however, neither contents are modified.

Flags S, P, AC are also modified in addition to Z and CY to reflect the results of the operation.

Example Register B contains data byte 62H and the accumulator contains data byte 57H. Compare the contents of register B with those of the accumulator.

Instruction: CMP B Hex Code: B8

Before instruction

A	57	XX	F
B	62	XX	C

After instruction

		CY	
A	57	1	F
B	62	XX	C

Flags: S = 1, Z = 0, AC = 1, P = 1, CY = 1

Results after executing the instruction:

- No contents are changed.
- Carry flag is set because (A) < (B).
- S, Z, P, AC flags will also be modified as listed above.

CPI: Compare Immediate with Accumulator

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
CPI	8-bit	2	2	7	FE

Description The second byte (8-bit data) is compared with the contents of the accumulator. The values being compared remain unchanged and the results of the comparison are indicated by setting the flags as follows.

- If (A) < Data: Carry flag is set and Zero flag is reset.
- If (A) = Data: Zero flag is set and Carry flag is reset.
- If (A) > Data: Carry and Zero flags are reset.

The comparison of two bytes is performed by subtracting the data byte from the contents of the accumulator; however, neither contents are modified.

Flags S, P, AC are also modified in addition to Z and CY to reflect the result of the operation.

Example Assume the accumulator contains data byte C2H. Compare 98H with the accumulator contents.

Instruction: CPI 98H Hex Code: FE 98

Results after executing the instruction:

- The accumulator contents remain unchanged.
- Z and CY flags are reset because (A) > Data.
- Other flags: S = 0, AC = 0, P = 0.

Example Compare data byte C2H with the contents of the accumulator in the above example.

Instruction: CPI C2H Hex Code: FE C2

Results after executing the instruction:

- The accumulator contents remain unchanged.
- Zero flag is set because (A) = Data.
- Other flags: S = 0, AC = 1, P = 1, CY = 0.

DAA: Decimal-Adjust Accumulator

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
DAA	None	1	1	4	27

Description The contents of the accumulator are changed from a binary value to two 4-bit binary-coded decimal (BCD) digits. This is the only instruction that uses the auxiliary flag (internally) to perform the binary-to-BCD conversion; the conversion procedure is described below.

Flags S, Z, AC, P, CY flags are altered to reflect the results of the operation. Instruction DAA converts the binary contents of the accumulator as follows:

1. If the value of the low-order four bits (D_3–D_0) in the accumulator is greater than 9 or if AC flag is set, the instruction adds 6 (06) to the low-order four bits.
2. If the value of the high-order four bits (D_7–D_4) in the accumulator is greater than 9 or if the Carry flag is set, the instruction adds 6 (60) to the high-order four bits.

Example Add decimal 12_{BCD} to the accumulator, which contains 39_{BCD}.

$$\begin{aligned} (A) = \quad 39_{BCD} &= 0\ 0\ 1\ 1 \quad 1\ 0\ 0\ 1 \\ +\ 12_{BCD} &= 0\ 0\ 0\ 1 \quad 0\ 0\ 1\ 0 \\ \hline 51_{BCD} &= 0\ 1\ 0\ 0 \quad 1\ 0\ 1\ 1 \\ & \qquad\quad 4 \qquad\quad B \end{aligned}$$

The binary sum is 4BH. The value of the low-order four bits is larger than 9. Add 06 to the low-order four bits.

$$\begin{aligned} 4B &= 0\ 1\ 0\ 0 \quad 1\ 0\ 1\ 1 \\ +\ 06 &= 0\ 0\ 0\ 0 \quad 0\ 1\ 1\ 0 \\ & \qquad\quad 1 \quad 1\ 1 \\ \hline 51 &= 0\ 1\ 0\ 1 \quad 0\ 0\ 0\ 1 \end{aligned}$$

Example Add decimal 68_{BCD} to the accumulator, which contains 85_{BCD}.

$$\begin{aligned} (A) = \quad 85_{BCD} &= 1\ 0\ 0\ 0 \quad 0\ 1\ 0\ 1 \\ +\ 68_{BCD} &= 0\ 1\ 1\ 0 \quad 1\ 0\ 0\ 0 \\ \hline 153_{BCD} &= 1\ 1\ 1\ 0 \quad 1\ 1\ 0\ 1 \end{aligned}$$

The binary sum is EDH. The values of both, low-order and high-order, four bits are higher than 9. Add 6 to both.

$$\begin{aligned} = \quad ED &= \quad 1\ 1\ 1\ 0 \quad 1\ 1\ 0\ 1 \\ +\ 66 &= \quad 0\ 1\ 1\ 0 \quad 0\ 1\ 1\ 0 \\ & \qquad 1\ 1 \quad 1 \quad 1 \\ \hline \boxed{1}53 &= \boxed{1}0\ 1\ 0\ 1 \quad 0\ 0\ 1\ 1 \\ CY & \quad\ CY \end{aligned}$$

The accumulator contains 53 and the Carry flag is set to indicate that the sum is larger than eight bits (153). The program should keep track of the Carry; otherwise it may be altered by the subsequent instructions.

DAD: Add Register Pair to H and L Registers

Opcode	Operand	Bytes	M-Cycles	T-States
DAD	Reg. pair	1	3	10

Hex Codes	
Reg. Pair	Hex
B	09
D	19
H	29
SP	39

Description The 16-bit contents of the specified register pair are added to the contents of the HL register and the sum is saved in the HL register. The contents of the source register pair are not altered.

Flags If the result is larger than 16 bits the CY flag is set. No other flags are affected.

Example Assume register pair HL contains 0242H. Multiply the contents by 2.

Instruction: DAD H **Hex Code:** 29

Before instruction	DAD operation	After instruction
H | 02 | 42 | L	0242 +0242 0484	H | 04 | 84 | L

Example Assume register pair HL is cleared. Transfer the stack pointer (register) that points to memory location 2099H to the HL register pair.

Instruction: DAD SP **Hex Code:** 39

Before instruction	DAD operation	After instruction
H | 00 | 00 | L	0000	H | 20 | 99 | L
SP | 2099 |	+2099	SP | 2099 |
	2099	

Note: After the execution of the instruction, the contents of the stack pointer register are not altered.

DCR: Decrement Source by 1

Opcode	Operand	Bytes	M-Cycles	T-States
DCR	Reg.	1	1	4
	Mem.	1	3	10

Reg.	Hex
B	05
C	0D
D	15
E	1D
H	25
L	2D
M	35
A	3D

Description The contents of the designated register/memory is decremented by 1 and the results are stored in the same place. If the operand is a memory location, it is specified by the contents of the HL register pair.

Flags S, Z, P, AC are modified to reflect the result of the operation. CY is not modified.

Example Decrement register B, which is cleared, and specify its contents after the decrement.

Instruction: DCR B Hex Code: 05

Before instruction

A	XX	F
B	00	XX C

Decrement operation

(B) = 0 0 0 0 0 0 0 0
−01 = 0 0 0 0 0 0 0 1

Subtraction is performed in 2's complement:

$$\begin{array}{rl} (B) = & 0\,0\,0\,0\ \ 0\,0\,0\,0 \\ & + \\ \text{2's complement of 1} = & \underline{1\,1\,1\,1\ \ 1\,1\,1\,1} \\ (B) = & 1\,1\,1\,1\ \ 1\,1\,1\,1 \end{array}$$

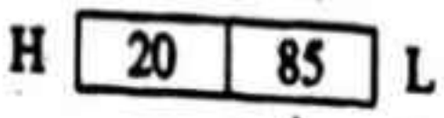
After the execution of the DCR instruction register B will contain FFH; however, this instruction does not modify the CY flag.

Example Decrement the contents of memory location 2085, which presently holds A0H.

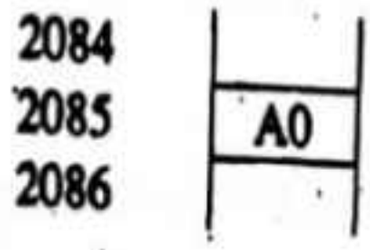
Assume the HL register contains 2085H.

Instruction: DCR M Hex Code: 35

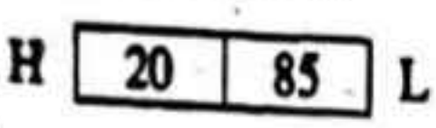
Before instruction

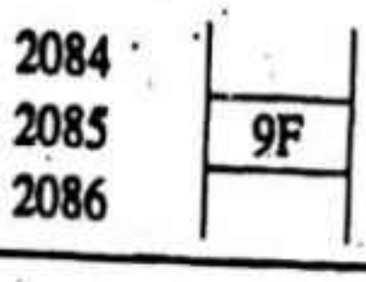


Memory



After instruction





DCX: Decrement Register Pair by 1

Opcode	Operand	Bytes	M-Cycles	T-States
DCX	Reg. pair	1	1	6

Hex Codes

Reg. Pair	Hex
B	0B
D	1B
H	2B
SP	3B

Description The contents of the specified register pair are decremented by 1. This instruction views the contents of the two registers as a 16-bit number.

Flags No flags are affected.

Example Register pair DE contains 2000H. Specify the contents of the entire register if it is decremented by 1.

Instruction: DCX D Hex Code: 1B

After subtracting 1 from the DE register pair the answer is

D | 1F | FF | E

Example Write instructions to set the Zero flag when a register pair (such as BC) is used as a down-counter.

To decrement the register pair, instruction DCX is necessary; instruction DCR is used for one register. However, instruction DCX does not set the Zero flag when the register pair goes to 0 and it continues counting indefinitely. The Zero flag can be set by using the following instructions.

For BC pair:

```
LOOP: DCX B       ;Decrement register pair BC
      MOV A,C     ;Load accumulator with the contents of register C
      ORA B       ;Set Zero flag if B and C are both 0
      JNZ LOOP    ;If Zero flag is not set, go back and decrement the contents of BC
                  pair
```

DI: Disable Interrupts

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
DI	None	1	1	4	F3

Description The Interrupt Enable flip-flop is reset and all the interrupts except the TRAP (8085) are disabled.

Flags No flags are affected.

Comments: This instruction is commonly used when the execution of a code sequence cannot be interrupted. For example, in critical time delays, this instruction is used at the beginning of the code and the interrupts are enabled at the end of the code. The 8085 TRAP cannot be disabled.

EI: Enable Interrupts

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
EI	None	1	1	4	FB

Description The Interrupt Enable flip-flop is set and all interrupts are enabled.

Flags No flags are affected.

Comments: After a system reset or the acknowledgment of an interrupt, the Interrupt Enable flip-flop is reset, thus disabling the interrupts. This instruction is necessary to reenable the interrupts (except TRAP).

HLT: Halt and Enter Wait State

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
HLT	None	1	2 or more	5 or more	76

Description The MPU finishes executing the current instruction and halts any further execution. The MPU enters the Halt Acknowledge machine cycle and Wait states are

inserted in every clock period. The address and the data bus are placed in the high impedance state. The contents of the registers are unaffected during the HLT state. An interrupt or reset is necessary to exit from the Halt state.

Flags No flags are affected.

IN: Input Data to Accumulator from a Port with 8-bit Address

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
IN	8-bit port address	2	3	10	DB

Description The contents of the input port designated in the operand are read and loaded into the accumulator.

Flags No flags are affected.

Comments: The operand is an 8-bit address; therefore, port addresses can range from 00H to FFH. While executing the instruction, a port address is duplicated on low-order (A_7–A_0) and high-order (A_{15}–A_8) address buses. Any one of the sets of address lines can be decoded to enable the input port.

INR: Increment Contents of Register/Memory by 1

Opcode	Operand	Bytes	M-Cycles	T-States
INR	Reg.	1	1	4
	Mem.	1	3	10

Hex Codes:

Reg.	Hex
B	04
C	0C
D	14
E	1C
H	24
L	2C
M	34
A	3C

Description The contents of the designated register/memory are incremented by 1 and the results are stored in the same place. If the operand is a memory location, it is specified by the contents of HL register pair.

Flags S, Z, P, AC are modified to reflect the result of the operation. CY is not modified.

Example Register D contains FF. Specify the contents of the register after the increment.

Instruction: INR D Hex Code: 14

```
(D) =    1 1 1 1  1 1 1 1
+ 1 =    0 0 0 0  0 0 0 1
       1 1 1 1 1  1 1 1     Carry
-------------------------
00 =  [0] 0 0 0 0  0 0 0 0
      CY
```

After the execution of the INR instruction, register D will contain 00H; however, no Carry flag is set.

Example Increment the contents of memory location 2075H, which presently holds 7FH. Assume the HL register contains 2075H.

Instruction: INR M Hex Code: 34

Before instruction

H [20 75] L

Memory	
2074	
2075	7F
2076	

After instruction

H [20 75] L

Memory	
2074	
2075	80
2076	

INX: Increment Register Pair by 1

Opcode	Operand	Bytes	M-Cycles	T-States
INX	Reg. pair	1	1	6

Hex Codes

Reg. Pair	Hex
B	03
D	13
H	23
SP	33

Description The contents of the specified register pair are incremented by 1. The instruction views the contents of the two registers as a 16-bit number.

Flags No flags are affected.

Example Register pair HL contains 9FFFH. Specify the contents of the entire register if it is incremented by 1.

Instruction: INX H Hex Code: 23
After adding 1 to the contents of the HL pair the answer is

H | A0 00 | L

JMP: Jump Unconditionally

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
JMP	16-bit	3	3	10	C3

Description The program sequence is transferred to the memory location specified by the 16-bit address. This is a 3-byte instruction; the second byte specifies the low-order byte and the third byte specifies the high-order byte.

Example Write the instruction at location 2000H to transfer the program sequence to memory location 2050H.

Instruction:

Memory Address	Code	Mnemonics
2000	C3	JMP 2050H
2001	50	
2002	20	

Comments: The 16-bit address of the operand is entered in memory in reverse order, the low-order byte first, followed by the high-order byte.

Jump Conditionally

Operand: 16-bit address

Op Code	Description	Flag Status	Hex Code	M-Cycles/T-States
JC	Jump on Carry	CY = 1	DA	2M/7T (if condition is not true)
JNC	Jump on No Carry	CY = 0	D2	3M/10T (if condition is true)
JP	Jump on positive	S = 0	F2	
JM	Jump on minus	S = 1	FA	
JPE	Jump on Parity Even	P = 1	EA	
JPO	Jump on Parity Odd	P = 0	E2	
JZ	Jump on Zero	Z = 1	CA	
JNZ	Jump on No Zero	Z = 0	C2	

23

Flags No flags are affected.

Comments: The 8085 requires only seven T-states when condition is not true. For example, instruction JZ 2050H will transfer the program sequence to location 2050H when the Zero flag is set (Z = 1) and the execution requires ten T-states. When the Zero flag is reset (Z = 0), the execution sequence will not be changed and this requires seven T-states.

LDA: Load Accumulator Direct

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
LDA	16-bit address	3	4	13	3A

Description The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator. The contents of the source are not altered. This is a 3-byte instruction; the second byte specifies the low-order address and the third byte specifies the high-order address.

Flags No flags are affected.

Example Assume memory location 2050H contains byte F8H. Load the accumulator with the contents of location 2050H.

Instruction: LDA 2050H Hex Code: 3A 50 20 (note the reverse order)

LDAX: Load Accumulator Indirect

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
LDAX	B/D reg. pair	1	2	7	

Reg.	Hex
BC	0A
DE	1A

Description 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.

Flags No flags are affected.

Example Assume the contents of register B = 20H, C = 50H, and memory location 2050H = 9FH. Transfer the contents of the memory location 2050H to the accumulator.

Instruction: LDAX B Hex Code: 0A

Register contents before instruction

A	XX	XX	F
B	20	50	C

Memory contents

2050 | 9F |

Register contents after instruction

A	9F	XX	F
B	20	50	C

LHLD: Load H and L Registers Direct

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
LHLD	16-bit address	3	5	16	2A

Description The instruction copies the contents of the memory location pointed out by the 16-bit address in register L and copies the contents of the next memory location in register H. The contents of source memory locations are not altered.

Flags No flags are affected.

Example Assume memory location 2050H contains 90H and 2051H contains 01H. Transfer memory contents to registers HL.

Instruction: LHLD 2050H Hex Code: 2A 50 20

Memory contents before instruction

2050	90
2051	01

Register contents after instruction

H | 01 | 90 | L

LXI: Load Register Pair Immediate

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
LXI	Reg. pair, 16-bit data	3	3	10	

Reg. Pair	Hex
B	01
D	11
H	21
SP	31

Description The instruction loads 16-bit data in the register pair designated in the operand. This is a 3-byte instruction; the second byte specifies the low-order byte and the third byte specifies the high-order byte.

Flags No flags are affected.

Example Load the 16-bit data 2050H in register pair BC.

Instruction: LXI B,2050H Hex Code: 01 50 20
This instruction loads 50H in register C and 20H in register B.

Comments: Note the reverse order in entering the code of 16-bit data. This is the only instruction that can directly load a 16-bit address in the stack pointer register.

MOV: Move—Copy from Source to Destination

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
MOV	Rd,Rs	1	1	4	See table below
MOV	M,Rs		2	7	
MOV	Rd,M				

Description This instruction copies the contents of the source register into the destination register; the contents of the source register are not altered. If one of the operands is a memory location, it is specified by the contents of HL registers.

Flags No flags are affected.

Hex Code

Destination Location	Source Location B	C	D	E	H	L	M	A
B	40	41	42	43	44	45	46	47
C	48	49	4A	4B	4C	4D	4E	4F
D	50	51	52	53	54	55	56	57
E	58	59	5A	5B	5C	5D	5E	5F
H	60	61	62	63	64	65	66	67
L	68	69	6A	6B	6C	6D	6E	6F
M	70	71	72	73	74	75		77
A	78	79	7A	7B	7C	7D	7E	7F

Example Assume register B contains 72H and register C contains 9FH. Transfer the contents of register C to register B.

Instruction: MOV B,C Hex Code: 41
Note the first operand B specifies the destination and the second operand C specifies the source.

Register contents before instruction

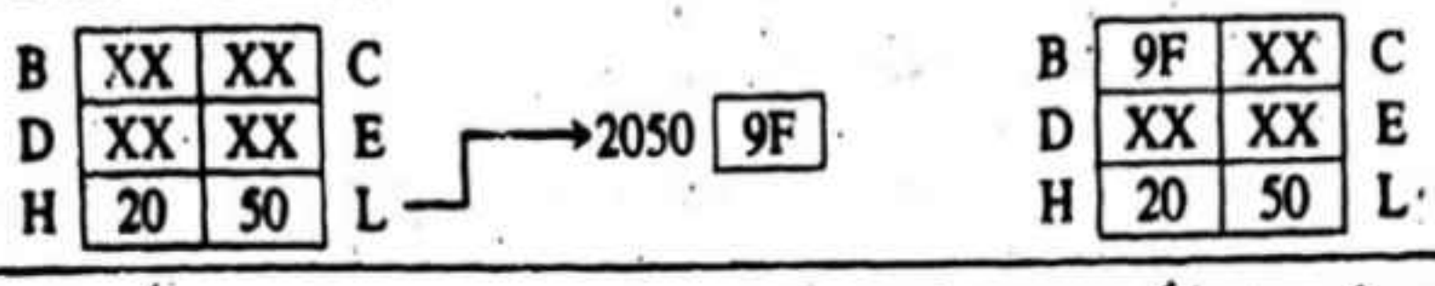
B	72	9F	C

Register contents after instruction

B	9F	9F	C

Example Assume the contents of registers HL are 20H and 50H, respectively. Memory location 2050H contains 9FH. Transfer the contents of the memory location to register B.

Instruction: MOV B,M Hex Code: 46



MVI: Move Immediate 8-Bit

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
MVI	Reg.,Data	2	2	7	
	Mem.,Data	2	3	10	

Reg.	Hex
B	06
C	0E
D	16
E	1E
H	26
L	2E
M	36
A	3E

Description The 8-bit data is stored in the destination register or memory. If the operand is a memory location, it is specified by the contents of HL registers.

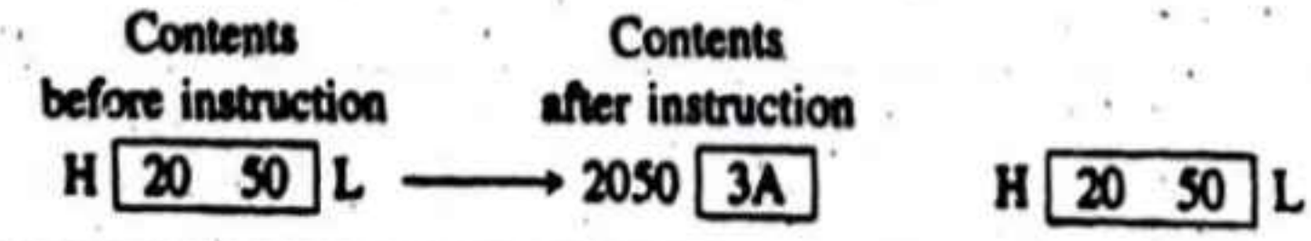
Flags No flags are affected.

Example Load 92H in register B.

Instruction: MVI B,92H Hex Code: 06 92
This instruction loads 92H in register B.

Example Assume registers H and L contain 20H and 50H, respectively. Load 3AH in memory location 2050H.

Instruction: MVI M,3AH Hex Code: 36



NOP: No Operation

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
NOP	None	1	1	4	00

Description No operation is performed. The instruction is fetched and decoded; however, no operation is executed.

Flags No flags are affected.

Comments: The instruction is used to fill in time delays or to delete and insert instructions while troubleshooting.

ORA: Logically OR with Accumulator

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
ORA	Reg.	1	1	4	
	Mem.	1	2	7	

Reg.	Hex
B	B0
C	B1
D	B2
E	B3
H	B4
L	B5
M	B6
A	B7

Description The contents of the accumulator are logically ORed with the contents of the operand (register or memory), and the results are placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers.

Flags Z, S, P are modified to reflect the results of the operation. AC and CY are reset.

Example Assume the accumulator has data byte 03H and register C holds byte 81H. Combine the bits of register C with the accumulator bits.

Instruction: ORA C Hex Code: B1

Register contents before instruction

A	03	XX	F
B	XX	81	C

Logical OR

03H = 0 0 0 0 0 0 1 1
81H = 1 0 0 0 0 0 0 1
82H = 1 0 0 0 0 0 1 0

S = 1, Z = 0, P = 0
Flags: CY = 0, AC = 0

Register contents after instruction

		S Z AC P CY	
A	83	1 0 0 0 0	F
B	XX	81	C

Comments: The instruction is commonly used to

- reset the CY flag by ORing the contents of the accumulator with itself.
- set the Zero flag when 0 is loaded into the accumulator by ORing the contents of the accumulator with itself.
- combine bits from different registers.

ORI: Logically OR Immediate

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
ORI	8-bit data	2	2	7	F6

Description The contents of the accumulator are logically ORed with the 8-bit data in the operand and the results are placed in the accumulator.

Flags S, Z, P are modifed to reflect the results of the operation. CY and AC are reset.

OUT: Output Data from Accumulator to a Port with 8-Bit Address

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
OUT	8-bit port address	2	3	10	D3

Description The contents of the accumulator are copied into the output port specified by the operand.

Flags No flags are affected.

Comments: The operand is an 8-bit address; therefore, port addresses can range from 00H to FFH. While executing the instruction, a port address is placed on the low-order address bus (A_7–A_0) as well as the high-order address bus (A_{15}–A_8). Any of the sets of address lines can be decoded to enable the output port.

PCHL: Load Program Counter with HL Contents

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
PCHL	None	1	1	6	E9

Description The contents of registers H and L are copied into the program counter. The contents of H are placed as a high-order byte and of L as a low-order byte.

Flags No flags are affected.

Comments: This instruction is equivalent to a 1-byte unconditional Jump instruction. A program sequence can be changed to any location by simply loading the H and L registers with the appropriate address and by using this instruction.

POP: Pop off Stack to Register Pair

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
POP	Reg. pair	1	3	10	

Reg.	Hex
B	C1
D	D1
H	E1
PSW	F1

Description The contents of the memory location pointed out by the stack pointer register are copied to the low-order register (such as C, E, L, and flags) of the operand. The stack pointer is incremented by 1 and the contents of that memory location are copied to the high-order register (B, D, H, A) of the operand. The stack pointer register is again incremented by 1.

Flags No flags are modified.

Example Assume the stack pointer register contains 2090H, data byte F5 is stored in memory location 2090H, and data byte 01H is stored in location 2091H. Transfer the contents of the stack to register pair H and L.

Instruction: POP H Hex Code: E1

Register contents before instruction

H	XX	XX	L
SP	2090		

Stack contents

090	F5
091	01
092	

Register contents after instruction

H	01	F5	L
SP	2092		

Comments: Operand PSW (Program Status Word) represents the contents of the accumulator and the flag register; the accumulator is the high-order register and the flags are the low-order register.

Note that the contents of the source, stack locations, are not altered after the POP instruction.

PUSH: Push Register Pair onto Stack

Opcode	Operand	Bytes	M-Cycles	T-States
PUSH	Reg. pair	1	3	12

Hex Code

Reg.	Hex
B	C5
D	D5
H	E5
PSW	F5

Description The contents of the register pair designated in the operand are copied into the stack in the following sequence. The stack pointer register is decremented and the contents of the high-order register (B, D, H, A) are copied into that location. The stack pointer register is decremented again and the contents of the low-order register (C, E, L, flags) are copied to that location.

Flags No flags are modified.

Example Assume the stack pointer register contains 2099H, register B contains 32H and register C contains 57H. Save the contents of the BC register pair on the stack.

Instruction: PUSH B Hex Code: C5

Register contents before instruction

B | 32 | 57 | C

SP | 2099 |

Stack contents after instruction

2097	57
2098	32
2099	XX

Register contents after instruction

B | 32 | 57 | C

SP | 2097 |

Comments: Operand PSW (Program Status Word) represents the contents of the accumulator and the flag register; the accumulator is the high-order register and the flags are the low-order register.

Note that the contents of the source registers are not altered after the PUSH instruction.

RAL: Rotate Accumulator Left through Carry

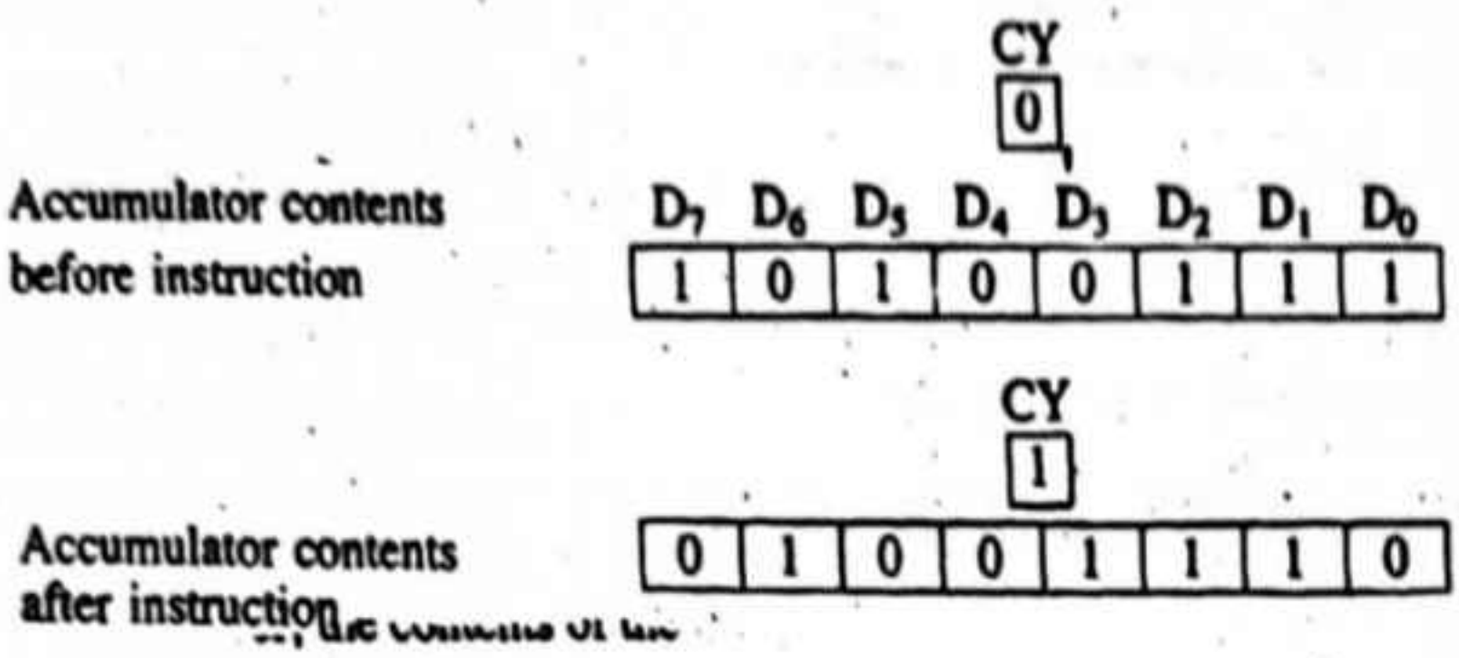
Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
RAL	None	1	1	4	17

Description Each binary bit of the accumulator is rotated left by one position through the Carry flag. Bit D_7 is placed in the bit in the Carry flag and the Carry flag is placed in the least significant position D_0.

Flags CY is modified according to bit D_7. S, Z, AC, P are not affected.

Example Rotate the contents of the accumulator through Carry, assuming the accumulator has A7H and the Carry flag is reset.

Instruction: RAL Hex Code: 17



Comment: This instruction effectively provides a 9-bit accumulator. The original contents of the accumulator can be restored by using instruction RAR (Rotate Accumulator Right through Carry). However; the contents will be modified if the instruction RRC (Rotate Accumulator Right) is used to restore the contents.

RAR: Rotate Accumulator Right through Carry

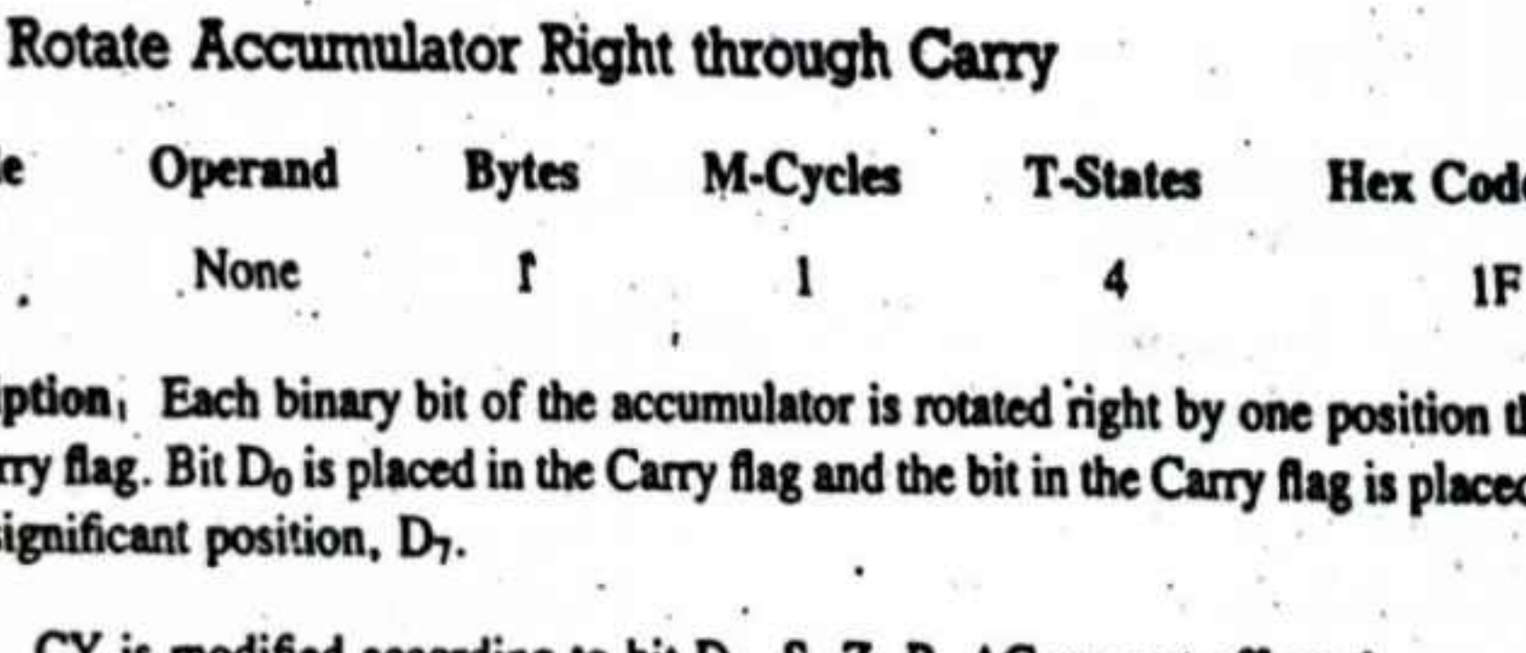
Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
RAR	None	1	1	4	1F

Description Each binary bit of the accumulator is rotated right by one position through the Carry flag. Bit D_0 is placed in the Carry flag and the bit in the Carry flag is placed in the most significant position, D_7.

Flags CY is modified according to bit D_0. S, Z, P, AC are not affected.

Example Rotate the contents of the accumulator assuming it contains A7H and the Carry flag is reset to 0.

Instruction: RAR Hex Code: 1F



RLC: Rotate Accumulator Left

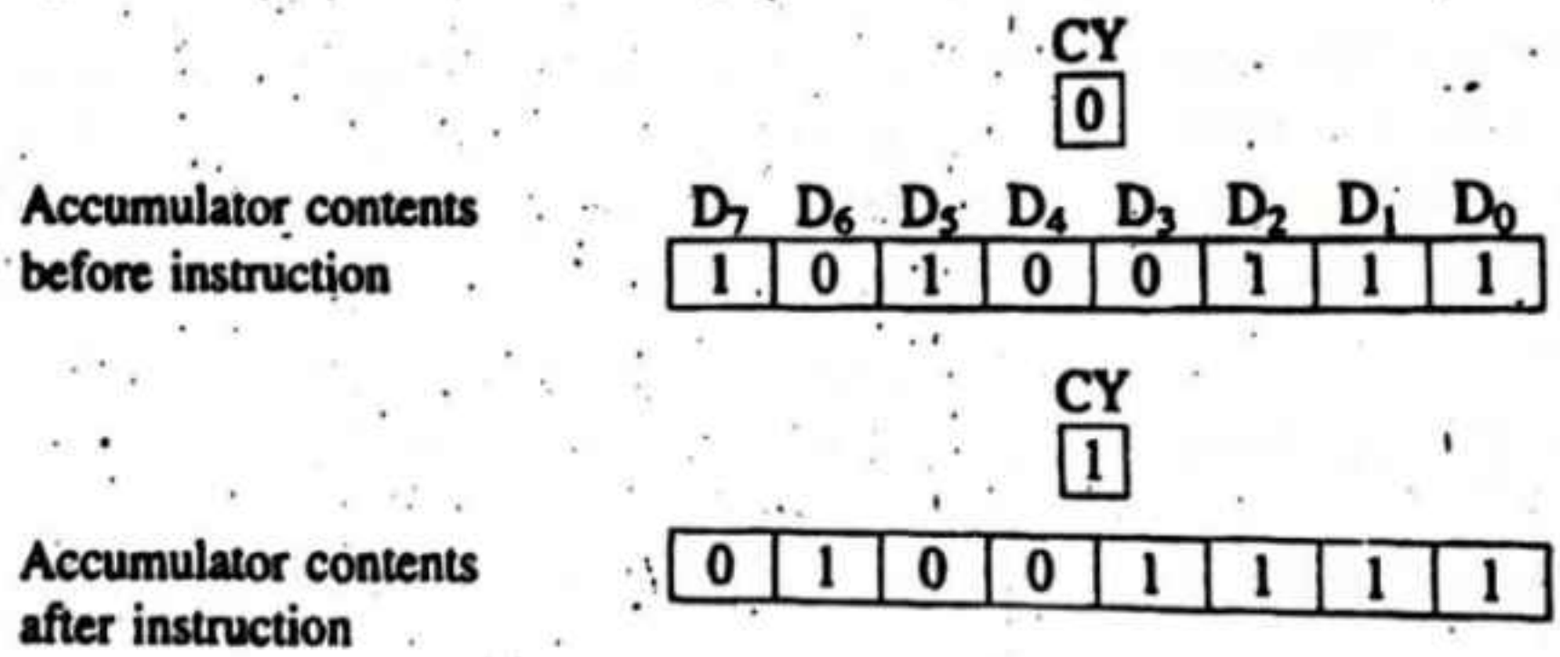
Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
RLC	None	1	1	4	07

Description Each binary bit of the accumulator is rotated left by one position. Bit D_7 is placed in the position of D_0 as well as in the Carry flag.

Flags CY is modified according to bit D_7. S, Z, P, AC are not affected.

Example Rotate the contents of the accumulator left, assuming it contains A7H and the Carry flag is reset to 0.

Instruction: RLC Hex Code: 07



Comments: The contents of bit D_7 are placed in bit D_0, and the Carry flag is modified accordingly. However, the contents of the Carry are not placed in bit D_0 as in instruction RAL.

RRC: Rotate Accumulator Right

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
RRC	None	1	1	4	0F

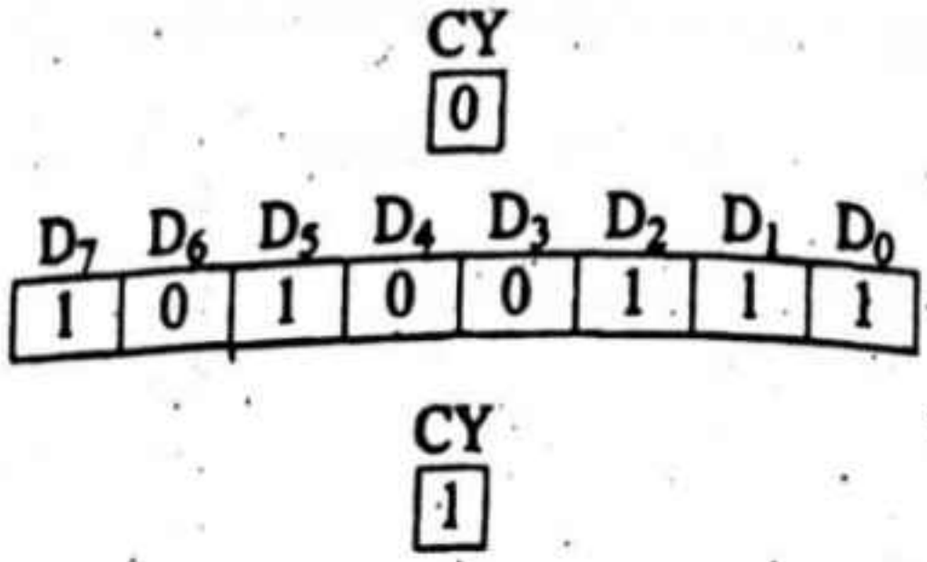
Description Each binary bit of the accumulator is rotated right by one position. Bit D_0 is placed in the position of D_7 as well as in the Carry flag.

Flags CY is modified according to bit D_0. S, Z, P, AC are not affected.

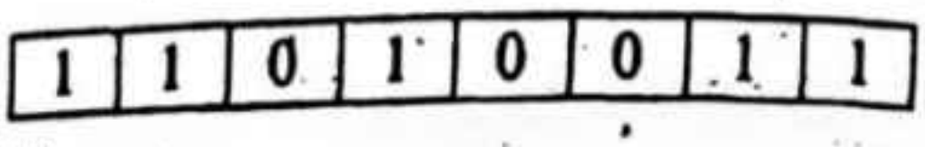
Example Rotate the contents of the accumulator right, if it contains A7H and the Carry flag is reset to 0.

Instruction: RRC Hex Code: 0F

Accumulator contents before instruction



Accumulator contents after instruction



Comments: The contents of bit D_0 are placed in bit D_7, and the Carry flag is modified accordingly. However, the contents of the Carry are not placed in bit D_7, as in the instruction RAR.

RET: Return from Subroutine Unconditionally

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
RET	None	1	3	10	C9

Description The program sequence is transferred from the subroutine to the calling program. The two bytes from the top of the stack are copied into the program counter and the program execution begins at the new address. The instruction is equivalent to POP Program Counter.

Flags No flags are affected.

Example Assume the stack pointer is pointing to location 2095H. Explain the effect of the RET instruction if the contents of the stack locations are as follows:

2095	50
2096	20

After instruction RET, the program execution is transferred to location 2050H and the stack pointer is shifted to location 2097H.

Comments: This instruction is used in conjunction with CALL or conditional call instructions.

Return Conditionally

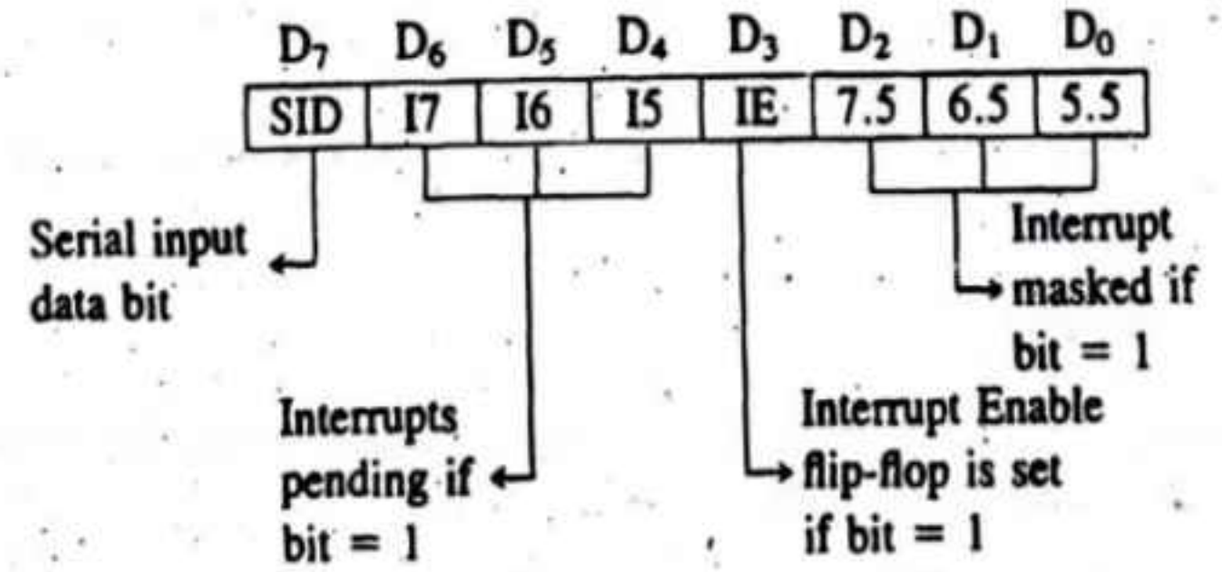
Op Code	Description	Flag Status	Hex Code	M-Cycles/T-States
RC	Return on Carry	CY = 1	D8	1/6 (if condition is not true)
RNC	Return with No Carry	CY = 0	D0	3/12 (if condition is true)
RP	Return on positive	S = 0	F0	*Note:* If condition is not true, it continues the sequence and thus requires fewer T-states.
RM	Return on minus	S = 1	F8	
RPE	Return on Parity Even	P = 1	E8	If condition is true, it returns to the calling program and thus requires more T-states.
RPO	Return on Parity Odd	P = 0	E0	
RZ	Return on Zero	Z = 1	C8	
RNZ	Return on No Zero	Z = 0	C0	

Flags No flags are affected.

RIM: Read Interrupt Mask

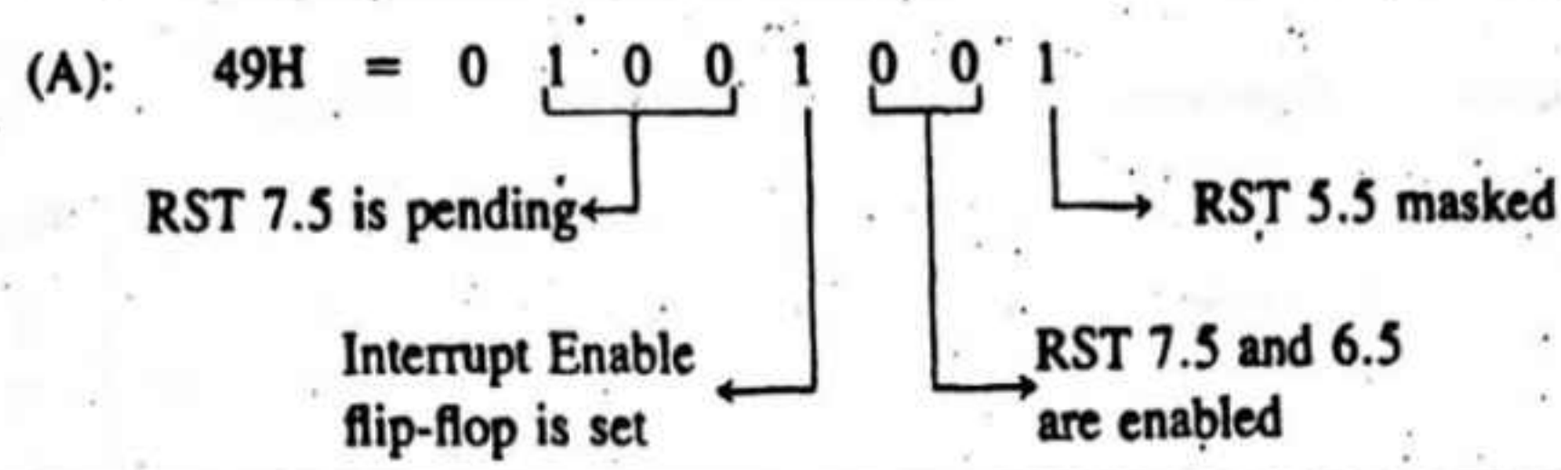
Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
RIM	None	1	1	4	20

Description This is a multipurpose instruction used to read the status of interrupts 7.5, 6.5, 5.5 and to read serial data input bit. The instruction loads eight bits in the accumulator with the following interpretations:



Flags No flags are affected.

Example After the execution of instruction RIM, the accumulator contained 49H. Explain the accumulator contents.



RST: Restart

Bytes	M-Cycles	T-States
1	3	12

Opcode/Operand	Binary Code	Hex Code	Restart Address (H)
RST 0	1 1 0 0 0 1 1 1	C7	0000
RST 1	1 1 0 0 1 1 1 1	CF	0008
RST 2	1 1 0 1 0 1 1 1	D7	0010
RST 3	1 1 0 1 1 1 1 1	DF	0018
RST 4	1 1 1 0 0 1 1 1	E7	0020
RST 5	1 1 1 0 1 1 1 1	EF	0028
RST 6	1 1 1 1 0 1 1 1	F7	0030
RST 7	1 1 1 1 1 1 1 1	FF	0038

Description The RST instructions are equivalent to 1-byte call instructions to one of the eight memory locations on page 0. The instructions are generally used in conjunction with interrupts and inserted using external hardware. However, these can be used as software instructions in a program to transfer program execution to one of the eight locations.

Flags No flags are affected.

Additional 8085 Interrupts The 8085 has four additional interrupts and these interrupts generate RST instructions internally and thus do not require any external hardware. These instructions and their Restart addresses are as follows:

Interrupts	Restart Address
TRAP	24H
RST 5.5	2CH
RST 6.5	34H
RST 7.5	3CH

SBB: Subtract Source and Borrow from Accumulator

Opcode	Operand	Bytes	M-Cycles	T-States
SBB	Reg.	1	1	4
	Mem.	1	2	7

Hex Code	
Reg.	Hex
B	98
C	99
D	9A
E	9B
H	9C
L	9D
M	9E
A	9F

Description The contents of the operand (register or memory) and the Borrow flag are subtracted from the contents of the accumulator and the results are placed in the accumulator. The contents of the operand are not altered; however, the previous Borrow flag is reset.

Flags All flags are altered to reflect the result of the subtraction.

Example Assume the accumulator contains 37H, register B contains 3FH, and the Borrow flag is already set by the previous operation. Subtract the contents of B with the borrow from the accumulator.

Instruction: SBB B Hex Code: 98

The subtraction is performed in 2's complement; however, the borrow needs to be added first to the subtrahend:

```
(B):                3F
Borrow:           + 1
Subtrahend:       40H = 0 1 0 0  0 0 0 0
2's complement of 40H = 1 1 0 0  0 0 0 0
(A)                   = 0 0 1 1  0 1 1 1
                      -------------------
Complement Carry:   0/ 1 1 1 1  0 1 1 1 = F7H
                    1/ 1 1 1 1  0 1 1 1
```

The Borrow flag is set to indicate the result is in 2's complement. The previous Borrow flag is reset during the subtraction.

SBI: Subtract Immediate with Borrow

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
SBI	8-bit data	2	2	7	DE

Description The 8-bit data (operand) and the borrow are subtracted from the contents of the accumulator, and the results are placed in the accumulator.

Flags All flags are altered to reflect the result of the operation.

Example Assume the accumulator contains 37H and the Borrow flag is set. Subtract 25H with borrow from the accumulator.

Instruction: SBI 25H Hex Code: DE 25

(Data): 25H
+ (Borrow): 1H
Subtrahend: $\overline{26H}$ = 0 0 1 0 0 1 1 0
2's complement of 26H = 1 1 0 1 1 0 1 0
(A) 37H = 0 0 1 1 0 1 1 1
1/ 0 0 0 1 0 0 0 1 = 11H
Complement Carry: 0/ 0 0 0 1 0 0 0 1 = 11H
Flags: S = 0, Z = 0, AC = 1
P = 1, CY = 0

SHLD: Store H and L Registers Direct

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
SHLD	16-bit address	3	5	16	22

Description The contents of register L are stored in the memory location specified by the 16-bit address in the operand, and the contents of H register are stored in the next memory location by incrementing the operand. The contents of registers HL are not altered. This is a 3-byte instruction; the second byte specifies the low-order address and the third byte specifies the high-order address.

Flags No flags are affected.

Example Assume the H and L registers contain 01H and FFH, respectively. Store the contents at memory locations 2050H and 2051H.

Instruction: SHLD 2050H Hex Code:: 22 50 20

Register contents before instruction

H | 01 | FF | L

Memory and register contents after instruction

2050	FF
2051	01

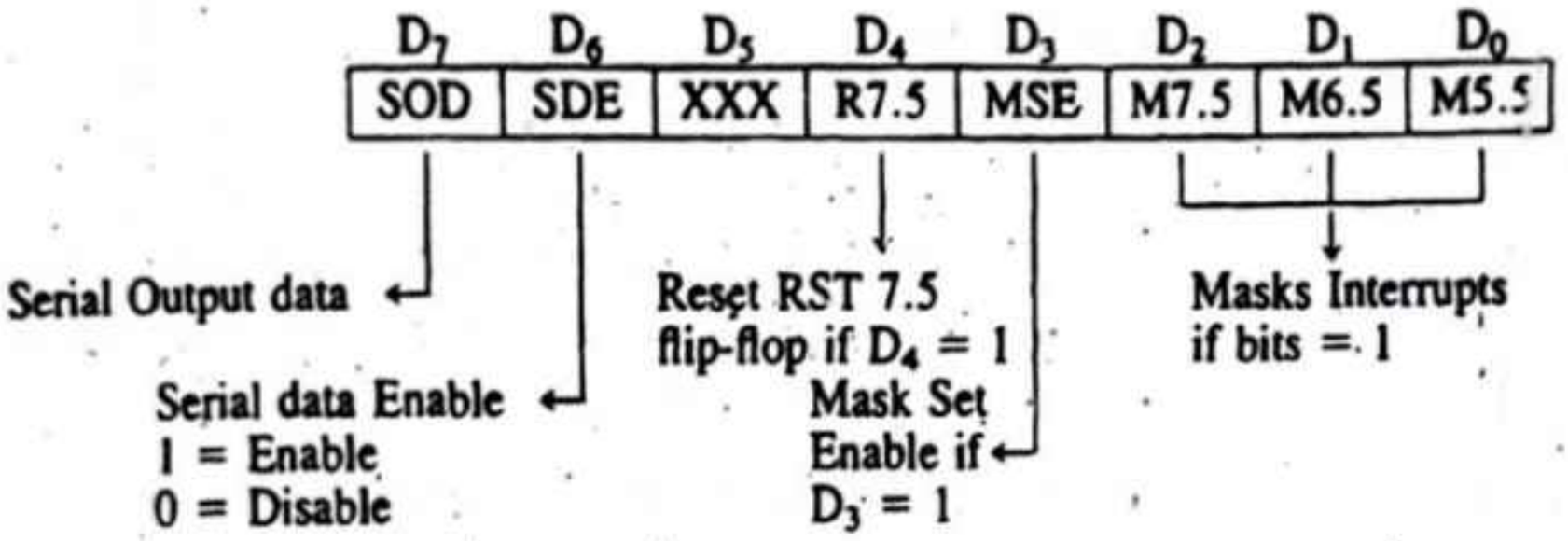
H | 01 | FF | L

SIM: Set Interrupt Mask

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
SIM	None	1	1	4	30

Description This is a multipurpose instruction and used to implement the 8085 interrupts (RST 7.5, 6.5, and 5.5) and serial data output.

The instruction interprets the accumulator contents as follows:



- SOD—Serial Output Data: Bit D_7 of the accumulator is latched into the SOD output line and made available to a serial peripheral if bit $D_6 = 1$.
- SDE—Serial Data Enable: If this bit = 1, it enables the serial output. To implement serial output, this bit needs to be enabled.
- XXX—Don't Care
- R7.5—Reset RST 7.5: If this bit = 1, RST 7.5 flip-flop is reset. This is an additional control to reset RST 7.5.
- MSE—Mask Set Enable: If this bit is high, it enables the functions of bits D_2, D_1, D_0. This is a master control over all the interrupt masking bits. If this bit is low, bits D_2, D_1, and D_0 do not have any effect on the masks.
- M7.5—D_2 = 0, RST 7.5 is enabled
  = 1, RST 7.5 is masked or disabled
- M6.5—D_1 = 0, RST 6.5 is enabled
  = 1, RST 6.5 is masked or disabled
- M5.5—D_0 = 0, RST 5.5 is enabled
  = 1, RST 5.5 is masked or disabled

Example Write instructions to enable interrupt RST 5.5 and mask other interrupts.

```
Instructions:  MVI A,0EH      ;Bits D3 = 1 and D0 = 0
               SIM            ;Enable RST 5.5
```

Example A TTY receiver line is connected to the SOD pin of the 8085. Disable all interrupts and send START bit (logic 0) to TTY without affecting interrupt masks.

Instructions: MVI A,40H ;D_7 = 0, START bit at logic 0
;D_6 = 1, Enables serial ouput bit D_7
;D_3 = 0, Does not affect masks
SIM ;Send START bit

Comments: This instruction does not affect TRAP interrupt.

SPHL: Copy H and L Registers to the Stack Pointer

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
SPHL	None	1	1	6 (8085) 5 (8080)	F9

Description The instruction loads the contents of the H and L registers into the stack pointer register; the contents of the H register provide the high-order address, and the contents of the L register provide the low-order address. The contents of the H and L registers are not altered.

Flags No flags are affected.

STA: Store Accumulator Direct

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
STA	16-bit	3	4	13	32

Description The contents of the accumulator are copied to a memory location specified by the operand. This is a 3-byte instruction; the second byte specifies the low-order address and the third byte specifies the high-order address.

Flags No flags are affected.

Example Assume the accumulator contains 9FH. Load the accumulator contents into memory location 2050H.

Instruction: STA 2050H Hex Code: 32 50 20

Register contents before instruction	Memory contents after instruction
A [9F XX] F	2050 [9F]

STAX: Store Accumulator Indirect

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
STAX	B/D reg. pair	1	2	7	Reg. Hex B 02 D 12

Description The contents of the accumulator are copied into the memory location specified by the contents of the operand (register pair). The contents of the accumulator are not altered.

Flags No flags are affected.

Example Assume the contents of the accumulator are F9H and the contents of registers B and C are 20H and 50H, respectively. Store the accumulator contents in memory location 2050H.

Instruction: STAX B Hex Code: 02

Register contents before instruction

A	F9	XX	F
B	20	50	C

2050 | F9 |

Register and memory contents after instruction

A	F9	XX	F
B	20	50	C

Comments: This instruction performs the same function as MOV A;M except this instruction uses the contents of BC or DE as memory pointers.

STC: Set Carry

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
STC	None	1	1	4	37

Description The Carry flag is set to 1.

Flags No other flags are affected.

SUB: Subtract Register or Memory from Accumulator

Opcode	Operand	Bytes	M-Cycles	T-States
SUB	Reg.	1	1	4
	Mem.	1	2	7

Hex Code	
Reg.	Hex
B	90
C	91
D	92
E	93
H	94
L	95
M	96
A	97

Description The contents of the register or the memory location specified by the operand are subtracted from the contents of the accumulator, and the results are placed in the accumulator. The contents of the source are not altered.

Flags All flags are affected to reflect the result of the subtraction.

Example Assume the contents of the accumulator are 37H and the contents of register C are 40H. Subtract the contents of register C from the accumulator.

Instruction: SUB C Hex Code: 91

```
          (C):  40H = 0 1 0 0  0 0 0 0
2's complement (C):   = 1 1 0 0  0 0 0 0
          (A):  37H = 0 0 1 1  0 1 1 1
                      ---------------------
                    0/ 1 1 1 1  0 1 1 1 = F7H
Complement Carry:   1/ 1 1 1 1  0 1 1 1
Flags: S = 1, Z = 0, AC = 0
       P = 0, CY = 1
```

The result, as a negative number, will be in 2's complement and thus the Carry (Borrow) flag is set.

SUI: Subtract Immediate from Accumulator

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
SUI	8-bit data	2	2	7	D6

Description The 8-bit data (the operand) are subtracted from the contents of the accumulator, and the results are placed in the accumulator.

Flags All flags are modified to reflect the results of the subtraction.

Example Assume the accumulator contains 40H. Subtract 37H from the accumulator.

Instruction: SUI 37H Hex Code: D6 37

```
Subtrahend: 37H =  0 0 1 1  0 1 1 1
2's complement of 37H =  1 1 0 0  1 0 0 1
                         +
(A): 40H =  0 1 0 0  0 0 0 0
            ---------------------
            1/ 0 0 0 0  1 0 0 1
Complement Carry:  0/ 0 0 0 0  1 0 0 1 = 09H
Flags: S = 0, Z = 0, AC = 0
       P = 1, CY = 0
```

XCHG: Exchange H and L with D and E

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
XCHG	None	1	1	4	EB

Description The contents of register H are exchanged with the contents of register D, and the contents of register L are exchanged with the contents of register E.

Flags No flags are affected.

XRA: Exclusive OR with Accumulator

Opcode	Operand	Bytes	M-Cycles	T-States
XRA	Reg.	1	1	4
	Mem.	1	2	7

Reg.	Hex
B	A8
C	A9
D	AA
E	AB
H	AC
L	AD
M	AE
A	AF

Description The contents of the operand (register or memory) are Exclusive ORed with the contents of the accumulator, and the results are placed in the accumulator. The contents of the operand are not altered.

Flags Z, S, P are altered to reflect the results of the operation. CY and AC_1 are reset.

Example Assume the contents of the accumulator are 77H and of register D are 56H. Exclusive OR the contents of the register D with the accumulator.

Instruction: XRA D Hex Code: AA

(A): 77H = 0 1 1 1 0 1 1 1
(D): 56H = 0 1 0 1 0 1 1 0
Exclusive OR: 0 0 1 0 0 0 0 1
Flags: S = 0, Z = 0, P = 1,
CY = 0, AC = 0

XRI: Exclusive OR Immediate with Accumulator

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
XRI	8-bit data	2	2	7	EE

Description The 8-bit data (operand) are Exclusive ORed with the contents of the accumulator, and the results are placed in the accumulator.

Flags Z, S, P are altered to reflect the results of the operation. CY and AC are reset.

Example Assume the contents of the accumulator are 8FH. Exclusive OR the contents of the accumulator with A2H.

Instruction: XRI A2H Hex Code: EE A2

(A): 8FH = 1 0 0 0 1 1 1 1
(Data): A2H = 1 0 1 0 0 0 1 0
Exclusive OR: 0 0 1 0 1 1 0 1
Flags: S = 0, Z = 0, P = 1
CY = 0, AC = 0

XTHL: Exchange H and L with Top of Stack

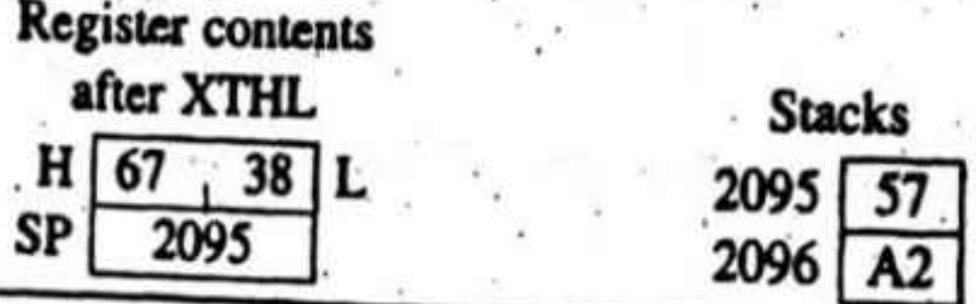
Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
XTHL	None	1	5	16	E3

Description The contents of the L register are exchanged with the stack location pointed out by the contents of the stack pointer register. The contents of the H register are exchanged with the next stack location (SP + 1); however, the contents of the stack pointer register are not altered.

Flags No flags are affected.

Example The contents of various registers and stack locations are as shown:

Illustrate the contents of these registers after instruction XTHL



8085 Instruction Summary: Hexadecimal Order

Hex	Mnemonic	Hex	Mnemonic	Hex	Mnemonic	Hex	Mnemonic
00	NOP	11	LXI D	21	LXI H	31	LXI SP
01	LXI B	12	STAX D	22	SHLD	32	STA
02	STAX B	13	INX D	23	INX H	33	INX SP
03	INX B	14	INR D	24	INR H	34	INR M
04	INR B	15	DCR D	25	DCR H	35	DCR M
05	DCR B	16	MVI D	26	MVI H	36	MVI M
06	MVI B	17	RAL	27	DAA	37	STC
07	RLC	19	DAD D	29	DAD H	39	DAD SP
09	DAD B	1A	LDAX D	2A	LHLD	3A	LDA
0A	LDAX B	1B	DCX D	2B	DCX H	3B	DCX SP
0B	DCX B	1C	INR E	2C	INR L	3C	INR A
0C	INR C	1D	DCR E	2D	DCR L	3D	DCR A
0D	DCR C	1E	MVI E	2E	MVI L	3E	MVI A
0E	MVI C	1F	RAR	2F	CMA	3F	CMG
0F	RRC	20	RIM	30	SIM	40	MOV B,B