Unit-4
Assembly Language program

Instruction Set of 8085

An **instruction** is a binary pattern designed inside a microprocessor to perform a specific function. The entire group of instructions that a microprocessor supports is called **Instruction Set**.

8085 has **246** instructions. Each instruction is represented by an 8-bit binary value. These 8- bits of binary value is called **Op-Code** or **Instruction Byte**.

Following are the classification of instructions:

a) Data Transfer Instruction
b) Arithmetic Instructions
c) Logical Instructions
d) Branching Instructions
e) Control Instructions

a) Data Transfer Instruction

These instructions move data between registers, or between memory and registers. These instructions copy data from source to destination. While copying, the contents of source are not modified.

Example: MOV, MVI

b) Arithmetic Instructions

These instructions perform the operations like addition, subtraction, increment and decrement.

Example: ADD, SUB, INR, DCR

c) Logical Instructions

These instructions perform logical operations on data stored in registers and memory. The logical operations are: AND, OR, XOR, Rotate, Compare and Complement.

Example: ANA, ORA, RAR, RAL, CMP, CMA

d) Branching Instructions

Branching instructions refer to the act of switching execution to a different instruction sequence as a result of executing a branch instruction. The three types of branching instructions are: **Jump, Call and Return.**

e) Control Instructions

The control instructions control the operation of microprocessor. **Examples: HLT, NOP, EI (Enable Interrupt), DI (Disable Interrupt).**

1. 8085 instruction set.

Sr.	Instruction	Description	ExampleDATA
DATA TRANSFER INSTRUCTIONS			
1.	MOV R_d, R_s MOV M, R_s MOV R_s, M	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, its location is specified by the contents of the HL registers.	MOV B, C MOV B, M
2.	MVI Rd, data MVI M, data	The 8-bit data is stored in the destination register or memory. If the operand is a memory location, its location is specified by the contents of the HL registers.	MVI B, 57H MVI M, 57H
3.	LDA 16-bit address	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.	LDA 2034H
4.	LDAX B/D Reg. pair	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.	LDAX B
5.	LXI Reg.-pair, 16-bit data	The instruction loads 16-bit data in the register pair designated in the operand.	LXI H, 2034H LXI H, XYZ
6.	LHLD 16-bit address	The instruction copies the contents of the memory location pointed out by the 16-bit address into register L and copies the contents of the next memory location into register H. The contents of source memory locations are not altered.	LHLD 2040H
7.	STA 16-bit address	The contents of the accumulator are copied into the 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.	STA 4350H
8.	STAX Reg. pair	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.	STAX B

Sr.	Instruction	Description	Example
9.	SHLD 16-bit address	The contents of register L are stored into the memory location specified by the 16-bit address in the operand and the contents of H register are stored into 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.	SHLD 2470H
10.	XCHG	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.	XCHG
11.	SPHL	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.	SPHL
12.	XTHL	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.	XTHL
13.	PUSH Reg. pair	The contents of the register pair designated in the operand are copied onto 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.	PUSH B PUSH A
14.	POP Reg. pair	The contents of the memory location pointed out by the stack pointer register are copied to the low-order register (C, E, L, status 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.	POP H POP A
15.	OUT 8-bit port address the	he contents of the accumulator T are copied into I/O port specified by the operand.	OUT F8H
16.	IN 8-bit port address the	The contents of the input port designated in operand are read and loaded into the accumulator.	IN 8CH

ARITHMETIC INSTRUCTIONS

Sr.	Instruction	Description	Example
17.	ADD R ADD M	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, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the addition.	ADD B ADD M
18.	ADC R ADC M	The contents of the operand (register or memory) and the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the addition.	ADC B ADC M
19.	ADI 8-bit data	The 8-bit data (operand) is added to the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the addition.	ADI 45H
20.	ACI 8-bit data	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. All flags are modified to reflect the result of the addition.	ACI 45H
21.	DAD Reg. pair	The 16-bit contents of the specified register pair are added to the contents of the HL register and the sum is stored in the HL register. The contents of the source register pair are not altered. If the result is larger than 16 bits, the CY flag is set. No other flags are affected.	DAD H
22.	SUB R SUB M	The contents of the operand (register or memory) are subtracted from the contents of the accumulator, and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the subtraction.	SUB B SUB M
23.	SBB R SBB M	The contents of the operand (register or memory) and the Borrow flag are subtracted from the contents of the accumulator and the result is placed in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the subtraction.	SBB B SBB M

Sr.	Instruction	Description	Example
24.	SUI 8-bit data	The 8-bit data (operand) is subtracted from the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the subtraction.	SUI 45H
25.	SBI 8-bit data	The 8-bit data (operand) and the Borrow flag are subtracted from the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the subtraction.	SBI 45H
26.	INR R INR M	The contents of the designated register or memory are incremented by 1 and the result is stored in the same place. If the operand is a memory location, its location is specified by the contents of the HL registers.	INR B INR M
27.	INX R	The contents of the designated register pair are incremented by 1 and the result is stored in the same place.	INX H
28.	DCR R DCR M	The contents of the designated register or memory are decremented by 1 and the result is stored in the same place. If the operand is a memory location, its location is specified by the contents of the HL registers.	DCR B DCR M
29.	DCX R	The contents of the designated register pair are decremented by 1 and the result is stored in the same place.	DCX H
30.	DAA	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 to perform the binary to BCD conversion, and the conversion procedure is described below. S, Z, AC, P, CY flags are altered to reflect the results of the operation. If the value of the low-order 4-bits in the accumulator is greater than 9 or if AC flag is set, the instruction adds 6 to the low-order four bits. If the value of the high-order 4-bits in the accumulator is greater than 9 or if the Carry flag is set, the instruction adds 6 to the high-order four bits.	DAA

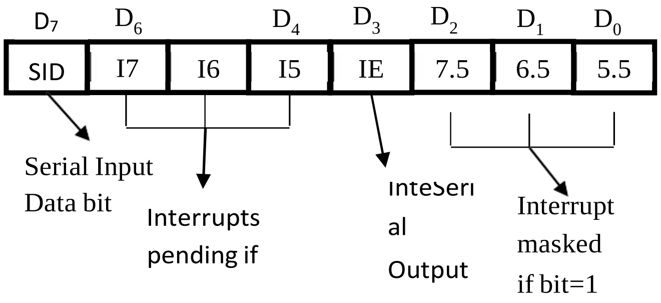
Sr.	**Instruction**	**Description**	**Example**
BRANCHING INSTRUCTIONS			
31.	JMP 16-bit address	The program sequence is transferred to the memory location specified by the 16-bit address given in the operand.	JMP 2034H JMP XYZ
Jump conditionally		*The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW as described below.*	
32.	JC 16-bit address	Jump on Carry, Flag Status: CY=1	JC 2050H
33.	JNC 16-bit address	**Jump on no Carry**, Flag Status: CY=0	JNC 2050H
34.	JP 16-bit address	Jump on positive, Flag Status: S=0	JP 2050H
35.	JM 16-bit address	Jump on minus, Flag Status: S=1	JM 2050H
36.	JZ 16-bit address	Jump on zero, Flag Status: Z=1	JZ 2050H
37.	JNZ 16-bit address	Jump on no zero, Flag Status: Z=0	JNZ 2050H
38.	JPE 16-bit address	Jump on parity even, Flag Status: P=1	JPE 2050H
39.	JPO 16-bit address	Jump on parity odd, Flag Status: P=0	JPO 2050H
40.	CALL 16-bit address	The program sequence is transferred to the memory location specified by the 16-bit address given in the operand. Before the transfer, the address of the next instruction after CALL (the contents of the program counter) is pushed onto the stack.	CALL 2034H CALL XYZ
Call conditionally		*The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW as described below. Before the transfer, the address of the next instruction after the call (the contents of the program counter) is pushed onto the stack.*	
41.	CC 16-bit address	Call on Carry, Flag Status: CY=1	CC 2050H
42.	CNC 16-bit address	Call on no Carry, Flag Status: CY=0	CNC 2050H
43.	CP 16-bit address	Call on positive, Flag Status: S=0	CP 2050H
44.	CM 16-bit address	Call on minus, Flag Status: S=1	CM 2050H
45.	CZ 16-bit address	Call on zero, Flag Status: Z=1	CZ 2050H
46.	CNZ 16-bit address	Call on no zero, Flag Status: Z=0	CNZ 2050H
47.	CPE 16-bit address	Call on parity even, Flag Status: P=1	CPE 2050H
48.	CPO 16-bit address	Call on parity odd, Flag Status: P=0	CPO 2050H

Sr.	Instruction	Description	Example
49.	RET	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 program execution begins at the new address.	RET
	Return from subroutine conditionally	*The program sequence is transferred from the subroutine to the calling program based on the specified flag of the PSW as described below. The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address.*	
50.	RC	Return on Carry, Flag Status: CY=1	RC
51.	RNC	Return on no Carry, Flag Status: CY=0	RNC
52.	RP	Return on positive, Flag Status: S=0	RP
53.	RM	Return on minus, Flag Status: S=1	RM
54.	RZ	Return on zero, Flag Status: Z=1	RZ
55.	RNZ	Return on no zero, Flag Status: Z=0	RNZ
56.	RPE	Return on parity even, Flag Status: P=1	RPE
57.	RPO	Return on parity odd, Flag Status: P=0	RPO
58.	PCHL	The contents of registers H and L are copied into the program counter. The contents of H are placed as the high-order byte and the contents of L as the low-order byte.	PCHL
59.	RST 0-7	The RST instruction is equivalent to a 1-byte call instruction to one of eight memory locations depending upon the number. 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. The addresses are: **Instruction** **Restart Address** RST 0 0000H RST 1 0008H RST 2 0010H RST 3 0018H RST 4 0020H RST 5 0028H RST 6 0030H RST 7 0038H	RST 3

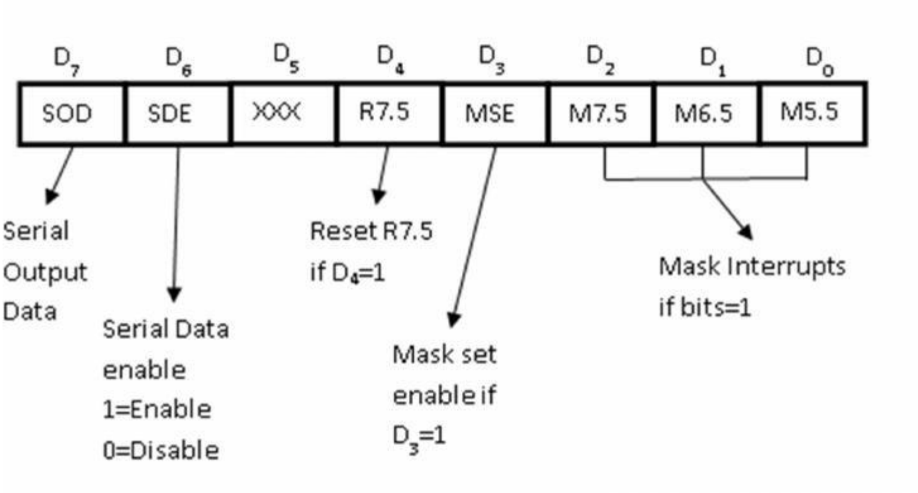
Sr.	Instruction	Description	Example
The 8085 has four additional interrupts and these interrupts generate RST instructions internally and thus do not require any external hardware.			
60.	TRAP	It restart from address 0024H	TRAP
61.	RST 5.5	It restart from address 002CH	RST 5.5
62.	RST 6.5	It restart from address 0034H	RST 6.5
63.	RST 7.5	It restart from address 003CH	RST 7.5
LOGICAL INSTRUCTIONS			
64.	CMP R CMP M	The contents of the operand (register or memory) are compared with the contents of the accumulator. Both contents are preserved. The result of the comparison is shown by setting the flags of the PSW as follows: if (A) < (reg/mem): carry flag is set if (A) = (reg/mem): zero flag is set if (A) > (reg/mem): carry and zero flags are reset	CMP B CMP M
65.	CPI 8-bit data	The second byte (8-bit data) is compared with the contents of the accumulator. The values being compared remain unchanged. The result of the comparison is shown by setting the flags of the PSW as follows: if (A) < data: carry flag is set if (A) = data: zero flag is set if (A) > data: carry and zero flags are reset	CPI 89H
66.	ANA R ANA M	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. S, Z, P are modified to reflect the result of the operation. CY is reset. AC is set.	ANA B ANA M
67.	ANI 8-bit data	The contents of the accumulator are logically ANDed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY is reset. AC is set.	ANI 86H
68.	XRA R XRA M	The contents of the accumulator are Exclusive ORed 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. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.	XRA B XRA M

Sr.	Instruction	Description	Example
69.	XRI 8-bit data	The contents of the accumulator are Exclusive ORed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.	XRI 86H
70.	ORA R ORA M	The contents of the accumulator are logically ORed 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. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.	ORA B ORA M
71.	ORI 8-bit data	The contents of the accumulator are logically ORed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.	ORI 86H
72.	RLC	Each binary bit of the accumulator is rotated left by one position. Bit D7 is placed in the position of D0 as well as in the Carry flag. CY is modified according to bit D7. S, Z, P, AC are not affected.	RLC
73.	RRC	Each binary bit of the accumulator is rotated right by one position. Bit D0 is placed in the position of D7 as well as in the Carry flag. CY is modified according to bit D0. S, Z, P, AC are not affected.	RRC
74.	RAL	Each binary bit of the accumulator is rotated left by one position through the Carry flag. Bit D7 is placed in the Carry flag, and the Carry flag is placed in the least significant position D0. CY is modified according to bit D7. S, Z, P, AC are not affected.	RAL
75.	RAR	Each binary bit of the accumulator is rotated right by one position through the Carry flag. Bit D0 is placed in the Carry flag, and the Carry flag is placed in the most significant position D7. CY is modified according to bit D0. S, Z, P, AC are not affected.	RAR
76.	CMA	The contents of the accumulator are complemented. No flags are affected.	CMA
77.	CMC	The Carry flag is complemented. No other flags are affected.	CMC
78.	STC	The Carry flag is set to 1. No other flags are affected.	STC
CONTROL INSTRUCTIONS			
79.	NOP	No operation is performed. The instruction is fetched and decoded. However no operation is executed.	NOP

Sr.	Instruction	Description	Example
80.	HLT	The CPU finishes executing the current instruction and halts any further execution. An interrupt or reset is necessary to exit from the halt state.	HLT
81.	DI	The interrupt enable flip-flop is reset and all the interrupts except the TRAP are disabled. No flags are affected.	DI
82.	EI	The interrupt enable flip-flop is set and all interrupts are enabled. No flags are affected. After a system reset or the acknowledgement of an interrupt, the interrupt enable flip-flop is reset, thus disabling the interrupts. This instruction is necessary to re enable the interrupts (except TRAP).	EI
83.	RIM	This is a multipurpose instruction used to read the status of interrupts 7.5, 6.5, 5.5 and read serial data input bit. The instruction loads eight bits in the accumulator with the following interpretations.	RIM



Sr.	Instruction	Description	Example
84.	SIM	This is a multipurpose instruction and used to implement the 8085 interrupts 7.5, 6.5, 5.5, and serial data output. The instruction interprets the accumulator contents as follows.	SIM



8085 Programs

1. **Program to add two 8-bit numbers.**

   Statement: *Add numbers 05H & 13H and display result in output port 03H.*

   ```
   MVI A,05H     //Move data 05H to accumulator
   MVI B,13H     //Move data 13H to B register
   ADD B         //Add contents of accumulator and B register
   OUT 03H       //Transfer result to output port 03H
   HLT           //Terminate the program.
   ```

   Input: A=05H B=13H **Output**: (port 03H) = 18H

2. **Program to add two 8-bit numbers.**

   Statement: *Add numbers from memory location 2050H & 2051H and store result in memory location 2055H.*

   ```
   LDA 2051H     //Load contents of memory location 2051 to accumulator
   MOV B,A       //Move contents of accumulator to B register
   LDA 2050H     //Load contents of memory location 2050 to accumulator
   ADD B         // Add contents of accumulator and B register
   STA 2055H     //Store contents of accumulator in memory location 2055H
   HLT           //Terminate the program.
   ```

   Input:

Memory Location	Data
2050H	45H
2051H	53H

   Output:

Memory Location	Data
2055H	98H

3. **Program to subtract two 8-bit numbers.**

   Statement: *Subtract numbers 25H & 12H and display result in output port 01H.*

   ```
   MVI A,25H     //Move data 05H to accumulator
   MVI B,12H     //Move data 13H to B register
   SUB B         //Add contents of accumulator and B register
   OUT 01H       //Transfer result to output port 01H
   HLT           //Terminate the program.
   ```

   Input: A=25H B=12H **Output**: (port 03H) = 13H

4. **Program to subtract two 8-bit numbers.**

   Statement: *Subtract numbers from memory location 2050H & 2051H and store result in memory location 2055H.*

   ```
   LDA 2051H     //Load contents of memory location 2051 to accumulator
   MOV B,A       //Move contents of accumulator to B register
   ```

```
LDA 2050H      //Load contents of memory location 2050 to accumulator
SUB B          // Add contents of accumulator and B register
STA 2055H      //Store contents of accumulator in memory location 2055H
HLT            //Terminate the program.
```

Input:

Memory Location	**Data**
2050H	65H
2051H	53H

Output:

Memory Location	**Data**
2055H	12H

5. **Program to find 1's complement of a number.**

Statement: *Input number from memory location 2013H and store result in memory location 2052H.*

```
LDA 2013H      //Load contents from memory location 2013H to accumulator
CMA            //Complement contents of accumulator
STA 2052H      //Store result in memory location 2052H
HLT            //Terminate the program.
```

Input:

Memory Location	**Data**
2013H	12H

Output:

Memory Location	**Data**
2052H	EDH

6. **Program to find 2's complement of a number.**

Statement: *Input number from memory location 2013H and store result in memory location 2052H.*

```
LDA 2013H      //Load contents from memory location 2013H to accumulator
CMA            //Complement contents of accumulator
ADI 01H        //Add 01H to the contents of accumulator
STA 2052H      //Store result in memory location 2052H
HLT            //Terminate the program.
```

Input:

Memory Location	**Data**
2013H	12H

Output:

Memory Location	**Data**
2052H	EEH

7. **Program to right shift 8-bit numbers.**

Statement*: Shift an eight-bit data four bits right. Assume data is in memory location 2051H. Store result in memory location 2055H.*

```
LDA 2051H      //Load data from memory location 2051H to accumulator
RAR            //Rotate accumulator 1-bit right
RAR
```

```
RAR
RAR
STA 2055H     //Store result in memory location 2055H
HLT           //Terminate the program.
```

8. **Program to left shift 8-bit numbers.**

Statement: *Shift an eight-bit data four bits left. Assume data is in memory location 2051H. Store result in memory location 2055H.*

```
LDA 2051H     //Load data from memory location 2051H to accumulator
RAL           //Rotate accumulator 1-bit left
RAR
RAR
RAR
STA 2055H     //Store result in memory location 2055H
HLT           //Terminate the program.
```

9. **Program to add two 16-bit numbers.**

Statement: *Add numbers 1124H & 2253H and store result in memory location 2055H & 2056H.*

```
LXI H,1124H   //Load 16-bit data 1124H to HL pair
LXI D,2253H   //Load 16-bit data 2253H to DE pair
MOV A,L       //Move contents of register L to Accumulator
ADD E         //Add contents of Accumulator and E register
MOV L,A       //Move contents of Accumulator to L register
MOV A,H       //Move contents of register H to Accumulator
ADC D         //Add contents of Accumulator and D register with carry
MOV H,A       //Move contents of Accumulator to register H
SHLD 2055H    //Store contents of HL pair in memory address 2055H & 2056H
HLT           //Terminate the program.
```

<u>Input:</u>

Register Pair	**Data**
HL	1124H
DE	2253H

<u>Output:</u>

Memory Location	**Data**
2055H	77H
2056H	33H

10. **Program to add two 16-bit numbers.**

Statement: *Input first number from memory location 2050H & 2051H and second number from memory location 2052H & 2053H and store result in memory location 2055H & 2056H.*

```
LHLD 2052H    //Load 16-bit number from memory location 2052H & 2053H to HL
pair
XCHG          //Exchange contents of HL pair and DE pair
LHLD 2050H    //Load 16-bit number from memory location 2050H & 2051H to HL
pair
MOV A,L       //Move contents of register L to Accumulator
ADD E         //Add contents of Accumulator and E register
MOV L,A       //Move contents of Accumulator to L register
MOV A,H      //Move contents of register H to Accumulator
ADC D        //Add contents of Accumulator and D register with carry
MOV H,A      //Move contents of Accumulator to register H
SHLD 2055H   //Store contents of HL pair in memory address 2055H & 2056H
HLT          //Terminate the program.
```

Input:

Memory Location	**Data**
2050H	33H
2051H	45H
2052H	24H
2053H	34H

Output:

Memory Location	**Data**
2055H	57H
2056H	79H

11. **Program to subtract two 16-bit numbers.**

Statement: *Subtract number 1234H from 4897H and store result in memory location 2055H & 2056H.*

```
LXI H,4567H  //Load 16-bit data 4897H to HL pair
LXI D,1234H  //Load 16-bit data 1234H to DE pair
MOV A,L      //Move contents of register L to Accumulator
SUB E        //Subtract contents of Accumulator and E register
MOV L,A      //Move contents of Accumulator to L register
MOV A,H      //Move contents of register H to Accumulator
SBB D        //Subtract contents of Accumulator and D register with borrow
MOV H,A      //Move contents of Accumulator to register H
SHLD 2055H   //Store contents of HL pair in memory address 2055H & 2056H
HLT          //Terminate the program.
```

Input:

Register Pair	**Data**
HL	4897H
DE	1234H

Output:

Memory Location	**Data**
2055H	63H
2056H	36H

12. **Program to subtract two 16-bit numbers.**

Statement: *Input first number from memory location 2050H & 2051H and second number from memory location 2052H & 2053H and store result in memory location 2055H & 2056H.*

```
LHLD 2052H //Load 16-bit number from memory location 2052H & 2053H to HL pair
XCHG         //Exchange contents of HL pair and DE pair
LHLD 2050H //Load 16-bit number from memory location 2050H & 2051H to HL pair
MOV A,L      //Move contents of register L to Accumulator
SUB E        //Subtract contents of Accumulator and E register
MOV L,A      //Move contents of Accumulator to L register
MOV A,H      //Move contents of register H to Accumulator
```

```
SBB D          //Subtract contents of Accumulator and D register with carry
MOV H,A        //Move contents of Accumulator to register H
SHLD 2055H     //Store contents of HL pair in memory address 2055H & 2056H
HLT            //Terminate the program.
```

Input:

Memory Location	**Data**
2050H	78H
2051H	45H
2052H	24H
2053H	34H

Output:

Memory Location	**Data**
2055H	54H
2056H	11H

13. Program to multiply two 8-bit numbers.

Statement: Multiply 06 and 03 and store result in memory location 2055H.

```
MVI A,00H
MVI B,06H
MIV C,03H
X: ADD B
DCR C
JNZ X
STA 2055H
HLT
```

14. Program to divide to 8-bit numbers.

Statement: Divide 08H and 03H and store quotient in memory location 2055H and remainder in memory location 2056H.

```
MVI A,08H
MVI B,03H
MVI C,00H
X: CMP B
   JC Y
   SUB B
   INR C
   JMP X
Y: STA 2056H
MOV A,C
STA 2055H
HLT
```

15. Program to find greatest among two 8-bit numbers.

Statement: Input numbers from memory location 2050H & 2051H and store greatest number in memory location 2055H.

```
LDA 2051H
MOV B,A
LDA 2050H
CMP B
JNC X
MOV A,B
X: STA 2055H
```

```
HLT
```

16. Program to find smallest among two 8-bit numbers.

Statement: Input numbers from memory location 2050H & 2051H and store smallest number in memory location 2055H.

```
LDA 2051H
MOV B,A
LDA 2050H
CMP B
JC X
MOV A,B
X: STA 2055H
HLT
```

17. Program to find whether a number is odd or even.

Statement: Input number from memory location 2050H and store result in 2055H.

```
LDA 2050H
ANI 01H
JZ X
MVI A,0DH
JMP Y
X: MVI A,0EH
Y: STA 2055H
HLT
```

18. Program to count no. of 1's in given number.

Statement: Input number from memory location 2050H and store result in 2055H.

```
LDA 2050H
MVI C,08H
MVI B,00H
X: RAR
    JNC Y
    INR B
Y: DCR C
    JNZ X
MOV A,B
STA 2055H
HLT
```

19. Display number from 1 to 10.

```
LXI H,2050H
MVI B,01H
MVI C,0AH
X: MOV M,B
   INX  H
   INR  B
   DCR C
   JNZ X
HLT
```

20. Find sum of numbers from 1 to 10.

```
LXI H,2050H
```

```
MVI B,01H
MVI C,0AH
MVI A,00H
X: ADD B
   INX  H
   INR  B
   DCR C
   JNZ X
STA 2055H
HLT
```

21. Display all odd numbers from 1 to 10.

```
LXI H,2050H
MVI B,01H
MVI C,0AH
X: MOV M,B
   INX H
   INR B

   INR  B
   DCR C
   DCR C
   JNZ X
HLT
```

22. Display all even numbers from 1 to 20.

```
LXI H,2050H
MVI B,02H
MVI C,14H
X: MOV M,B
   INX  H
   INR  B
   INR  B
   DCR C
   DCR C
   JNZ X
HLT
```

23. Display all even numbers from 10 to 50.

```
LXI H,2050H
MVI B,0AH
MVI C,32H
X: MOV M,B
   INX  H
   INR  B
   INR  B
   DCR C
   DCR C
   JNZ X
HLT
```

24. Find sum of 10 numbers in array.

```
LXI H,2050H
 MVI C,0AH
 MVI A,00H
```

```
X: MOV B,M
     ADD B
    INX  H
    DCR C
    JNZ X
STA 2060H
HLT
```

25. Find the largest element in a block of data. The length of the block is in the memory location 2200H and block itself starts from memory location 2201H. Store the maximum number in memory location 2300H.

```
LDA 2200H
MOV C,A
LXI H,2201
MVI A,00H
X: CMP M
   JNC Y
   MOV A,M
Y:  INX H
   DCR C
   JNZ X
 STA 2300H
HLT
```

26. Find smallest number in array.

```
LDA 2200H
MOV C,A
LXI H,2201H
MVI A,00H
X: CMP M
   JC Y
   MOV A,M
Y: INX H
   DCR C
   JNZ X
 STA 2300H
HLT
```

27. Generate Fibonacci series upto 10th term.

```
LXI H,2050H
MVI C,08H
MVI B,00H
MVI D,01H
MOV M,B
INX H
MOV M,D
X: MOV A,B
   ADD D
    MOV B,D
    MOV D,A
    INX H
    MOV M,A
    DCR C
    JNZ X
HLT
```

28. **Sort 10 numbers in ascending order in array.**

```
MVI C,0AH
DCR C
X: MOV D,C
    LXI H,2050H

Y: MOV A,M
    INX H
    CMP M
    JC Z

MOV B,M
MOV M,A
DCX H
MOV M,B
INX H

Z: DCR D
    JNZ Y
    DCR C
    JNZ X
HLT
```

29. **Sort numbers in descending order in array. Length of array is in memory location 2050H.**

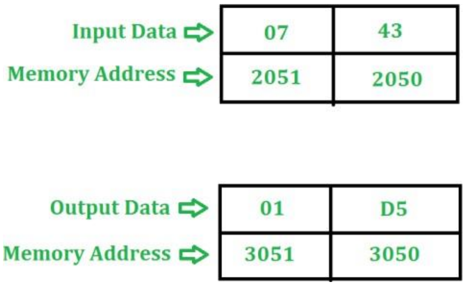
```
LDA 2050H
MVI C,A
DCR C
X: MOV D,C
    LXI H,2051H

Y: MOV A,M
    INX H
    CMP M
    JNC Z

MOV B,M
MOV M,A
DCX H
MOV M,B
INX H

Z: DCR D
    JNZ Y
    DCR C
    JNZ X
HLT
```

30. Multiply two 8 bit numbers 43H & 07H. Result is stored at address 3050 and 3051.

```
LXI H,0000H
MVI D,00H MVI
E,43H MVI
C,07H X: DAD D
   DCR C JNZ
  X
SHLD 2050H HLT
```

31. Multiply two 8 bit numbers stored at address 2050 and 2051. Result is stored at address 3050 and 3051.

```
LDA 2050H
MOV E,A LDA
2051H MOV C,A
MVI D,00H LXI
H,0000H X: DAD
D
   DCR C JNZ
  X
SHLD 3050H HLT
```



Input Data ⇨	07	43
Memory Address ⇨	2051	2050

Output Data ⇨	01	D5
Memory Address ⇨	3051	3050

8086 Programs

1. **Program to add two 8-bit numbers.**
   Statement: Add data 05H & 13H and store result in memory location 2050H.

```
MOV AL,05H
MOV BL,13H ADD
AL,BL MOV
2050H,AL HLT
```

2. **Program to add two 8-bit numbers.**

Statement: Input numbers from memory location 2050H and 2051H and store in memory location 2055H.

```
MOV SI,2050H
MOV AL,[SI] INC SI
MOV BL,[SI] ADD
AL,BL MOV
2055H,AL HLT
```

3. **Program to add two 16-bit numbers.**

Statement: Add numbers 1122H & 2233H and store result in memory location 2055H.

```
MOV AX,1122H
MOV BX,2233H
ADD AX,BX MOV
2055H,AH MOV
2056H,AL HLT
```

4. **Program to subtract two 8-bit numbers.**

Statement: Subtract data 05H from 13H and store result in memory location 2050H.

```
MOV AL,13H
MOV BL,05H SUB
AL,BL MOV
2050H,AL HLT
```

5. **Program to subtracct two 8-bit numbers.**

Statement: Input numbers from memory location 2050H and 2051H and store in memory location 2055H.

```
MOV SI,2050H
MOV AL,[SI] INC SI
MOV BL,[SI]
SUB AL,BL

MOV        2055H,AL
HLT
```

6. **Program to subtract two 16-bit numbers.**

Statement: Subtract numbers 1122H from 2233H and store result in memory location 2055H.

```
MOV AX,2233H
MOV BX,1122H
SUB AX,BX MOV
2055H,AH MOV
2056H,AL HLT
```

7. Program to multiply two 8-bit numbers.

Statement: Multiply 06H & 03H and store result in memory location 2055H.

```
MOV AL,06H
MOV BL,03H
MUL BL
MOV 2055H,AL HLT
```

8. Program to multiply two 8-bit numbers.

Statement: Multiply 43H & 13H and store result in memory location 2055H and 2056H. (This program works for 16-bit too.)

```
MOV       AX,0043H
MOV       BX,0013H
MUL BX
MOV       2055H,AH
MOV 2056H,AL HLT
```

9. Program to divide two 8-bit numbers.

Statement: Divide 43H & 13H and store result in memory location 2055H. MOV

```
AL,43H
MOV BL,13H DIV
BL
MOV 2055H,AL HLT
```

10. Program to divide two 8-bit numbers.

Statement: Divide 43H & 13H and store quotient in 2055H and remainder in 2056H.

```
MOV AL,43H
MOV        BL,13H
MOV   CL,00H   X:
CMP AL,BL
    JNC      Y   SUB
    AL,BL

    INC CL
Y: MOV 2056H,AL
   MOV AL,BL MOV
   2055H,AL
HLT
```

11. Program to divide two 16-bit numbers.

Statement: Divide 1243H & 0013H and store result in memory location 2055H & 2056H.

```
MOV       AX,1243H
MOV BX,0013H DIV
BX
MOV       2055H,AL
MOV        2056H,BL
HLT
```

12. Program to find sum of numbers from 1 to 10.

```
MOV        AL,00H
MOV        BL,01H
MOV   CL,0AH   X:
ADD BL
    INC          BL
    LOOP X
MOV        2055H,AL
HLT
```

13. Program to display numbers from 1 to 20.

```
MOV SI,2050H‘
MOV BL,01H MOV
CL,14H X: MOV
[SI],BL
    INC BL
    LOOP X
HLT
```

14. Program to find factorial of given number. (Number is in memory location 2050H).

```
MOV CX,2050H
MOV AX,00H X:
MUL CX
    LOOP X MOV
2055H,AH MOV
2056H,AL HLT
```

(Likewise all programs done in 8085 can be done in 8086)

15. Program to display string “I love my country” in screen.

```
.DATA
MESSAGE DB "I love my country$"
.CODE
START:
  MOV AX,DATA
  MOV DS,AX

  MOV AH,09H INT
  21H
  MOV AH,4CH INT
  21H

  END START
```

16. Program to display string “I love my country” in screen character by character.

```
.DATA
   MESSAGE DB "I love my country$"
.CODE
START:
    MOV AX,DATA
    MOV DS,AX

    LEA SI,MESSAGE
    MOV CL,11H
    L1:MOV DX,[SI]
      MOV AH,02H INT
     21H
     INC SI LOOP
     L1

    MOV AH,4CH INT
    21H
END START
```

17. Program to reverse any string.

```
.DATA
   MESSAGE DB "BSC CSIT$"
.CODE

START:
    MOV AX,DATA
    MOV DS,AX
    LEA SI,MESSAGE
    MOV CL,08H
    L1:MOV BX,[SI]
      PUSH BX INC
      SI LOOP L1

    MOV CL,05H L2:POP
    DX
      MOV AH,02H INT
      21H LOOP L2

    MOV AH,4CH INT
    21H
END START
```

(This program can be used for example program for stact PUSH and POP operation)

Some Exam Questions:

1) Write a program in 8-bit microprocessor to multiply two 16-bit numbers and store in the memory location starting from 3500H. Save the carry bits in the location starting from 3600H

2) Write an assembly language program to find the greatest number in an array in using 8 bit microprocessor. (Assume appropriate array data and address where minimum array size of 20 should be considered.)

3) Write an assembly language program to find the smallest number in an array using 8 bit microprocessor. (Assume appropriate array data and address where minimum array size of 15 should be considered.)

4) Ten number of 8-bit data stored at memory location 6000H. Write a program for 8085 microprocessor to calculate the sum of odd numbers and store the sum of odd numbers and store the sum at 6010H.(The sum may exceed 8-biys).

5) Write an assembly language program to subtract two 16-bit numbers.

6) Write and explain assembly language program to multiply 05H and 06H.

7) Write an ALP for 8086 to read string and print it in the reverse order.