

Practical – 6

Aim: Write an Assembly Language Program to perform the subtraction of two 8-bit numbers.

SUB Subtract the content of a register or a memory location from the content of accumulator and the result is stored in the accumulator.			
<p>→ SUB is a mnemonic that stands for ‘SUBTRACT’ contents of R from Accumulator. Here R stands for any of the following registers, or memory location M pointed by HL pair.</p> <div style="border: 1px solid #ccc; padding: 5px; margin: 10px 0;"> $R = A, B, C, D, E, H, L, \text{ or } M$ </div> <p>→ SUB B (It subtracts the content of B register from the content of the accumulator.) → SUB M (It subtracts the content of memory location pointed by HL pair from the content of accumulator)</p>			
SUB	R	$A = A - R$	SUB B
SUB	M	$A = A - M$	SUB 2050
<p>→ 1 byte instruction → In this instruction content of the register R will get subtracted from the Accumulator and the resultant difference will be stored on the Accumulator replacing the previous content of the Accumulator. → Where R can be any of the eight values, so as a result there are eight opcode for this type of instruction as mentioned in the above table. It occupies only 1-Byte in the memory.</p>			

MOV: Copy the data from one place to another.
<p>MOV Rd, Rs → Copies the content of Rs to Rd (MOV 1-byte instruction)</p>
<p>MOV M, Rs → MOV M, r will copy 8-bit value from the register r to the memory location as pointed by HL register pair. This instruction uses register addressing for specifying the data. → As “r” can have any one of the seven values –</p> <div style="border: 1px solid #ccc; padding: 5px; margin: 10px 0;"> $r = A, B, C, D, E, H, \text{ or } L$ </div> <p>→ Copies the content of register Rs to memory location pointed by HL Register</p>

- 2-byte instruction
- Copy the data byte from the register in to memory specified by the address in HL register.
- Thus, there are seven opcodes for this type of instruction. It occupies only 1-Byte in memory.

Mnemonics, Operand	Bytes
MOV M, A	1
MOV M, B	1
MOV M, C	1
MOV M, D	1
MOV M, E	1
MOV M, H	1
MOV M, L	1

MOV Rd, M

- Copies the content of memory location pointed by the HL register to the register Rd.
- 2 byte instruction
- Copy the data byte in to the register from the memory specified by the address in HL register.
- **MOV r, M** is an instruction where the 8-bit data content of the memory location as pointed by HL register pair will be moved to the register r. Thus this is an instruction to load register r with the 8-bit value from a specified memory location whose 16-bit address is in HL register pair.
- As r can have any of the seven values, there are seven opcodes for this type of instruction.

r = A, B, C, D, E, H, or L

Mnemonics, Operand	Bytes
MOV A, M	1
MOV B, M	1
MOV C, M	1
MOV D, M	1
MOV E, M	1
MOV H, M	1

MOV L, M

1

Examples:

MVI B, 10h

MOV A, B

MOV M, B

MOV C, M

MVI: Move immediate data to a register or memory location.

- **MVI** is a mnemonic, which actually means “Move Immediate”. With this instruction, we can load a register with an 8-bit or 1-Byte value.
- This instruction supports immediate addressing mode for specifying the data in the instruction.
- In the instruction “d8” stands for any 8-bit data, and ‘r’ stands for any one of the registers e.g. A, B, C, D, E, H or L. So this r can replace any one of the seven registers.
- As ‘r’ can have any of the seven register names, so there are seven opcodes for this type of instruction. It occupies 2-Bytes in the memory.

Mnemonics, Operand	Bytes
MVI A, Data	2
MVI B, Data	2
MVI C, Data	2
MVI D, Data	2
MVI E, Data	2
MVI H, Data	2
MVI L, Data	2

MVI Rd, #30H

- 30h is stored in register Rd

MVI M, #30H

- 30h is stored in memory location pointed by HL Reg
- 2 byte instruction

Examples:

MVI B, 10H [Loads the 8 bits of the 2nd byte in to the register specified]

MVI M, 30H

LDA: Load Accumulator

(This instruction copies the data from a given 16 bit address to the Accumulator)

- **LDA** is a mnemonic that stands for Load Accumulator with the contents from memory.
- In this instruction Accumulator will get initialized with 8-bit content from the 16-bit memory address as indicated in the instruction as a 16.

- This instruction uses absolute addressing for specifying the data.
- It occupies 3-Bytes in the memory.
- First Byte specifies the opcode, and the successive 2-Bytes provide the 16-bit address, i.e. 1-Byte each for each memory location.
- 3 byte instruction

LDA 3000H

- Load the data byte from Memory into Accumulator specified by 16 bit address
- Content of memory location 3000h is copied in accumulator

Example:1

start: nop

LDA var1

hlt

var1: db 04h

Example:2 (Store 45H data to Specific Memory Location 000BH)

start: nop

lxi h, 000Bh

MVI M, 45H

LDA 000Bh

Hlt

LXI: (Load register pair immediate)

The instruction loads 16-bit data in the register pair designated in the operand.

LXI Rp, 16 bit

- 3 byte instruction
- Load intermediate immediate 16 bit no. in a register pair
- These instructions are used to load the **16-bit** address into the register pair.
- We can use this instruction to load data from memory location using the memory address, which is stored in the register pair **rp**.
- The rp can be BC, DE, HL or SP.
- The LXI instructions and their Hex-codes are as follows.

Mnemonics, Operand	Bytes
LXI B	3
LXI D	3
LXI H	3
LXI SP	3

Example:

LXI B,2050H

STA

The content of accumulator is copied into the memory location.

STA 16 BIT

- **STA** is a mnemonic that stands for Store Accumulator contents in memory.
- In this instruction, Accumulator 8-bit content will be stored to a memory location whose 16-bit address is indicated in the instruction as a16.
- This instruction uses absolute addressing for specifying the destination.
- This instruction occupies 3-Bytes of memory.
- First Byte is required for the opcode, and next successive 2-Bytes provide the 16-bit address divided into 8-bits each consecutively.
- 3 byte instruction
- Load the data byte from A into the memory specified by 16 bit address

STA 2060H

Example:

start: nop

MVI A, 45h

STA 000Bh

Hlt

INX Rp

INX is a mnemonic that stands for “Increment extended register” and **rp** stands for register pair. And it can be any one of the following register pairs.

rp = BC, DE, or HL

- 2 byte instruction
- This instruction will be used to add 1 to the present content of the rp. And thus the result of the incremented content will remain stored in rp itself.
- Though it is an arithmetic instruction, note that, flag bits are not at all affected by the execution of this instruction.
- A register pair is generally used to store 16-bit memory address.
- If flag bits got affected during increment of a memory address, then it may cause problems in many cases.
- So as per design of 8085, flag bits are not getting affected by the execution of this instruction **INXrp**.
- As rp can have any one of the three values, there are three opcodes for this type of instruction. It occupies only 1-Byte in memory.

Mnemonics, Operand	Bytes
INX B	1
INX D	1
INX H	1

→ Example:
start: nop
LXI B, 4523h
INX B
Hlt

Opcode	Operand	Meaning	Explanation
SUB	R M	Subtract the register or the memory from the accumulator	The contents of the register or the memory are subtracted from the contents of the accumulator, and the result is stored in the accumulator. Example – SUB K
INX	R	Increment register pair by 1	The contents of the designated register pair are incremented by 1 and their result is stored at the same place. Example – INX K

For subtraction of two 8-bit numbers :

; <Program title>

jmp start

;data

;code

start: nop

MVI A,7H

MVI B,2H

SUB B

Hlt

Output:

Registers			Flag	
<i>A</i>	05		<i>S</i>	0
<i>BC</i>	02	00		
<i>DE</i>	00	00	<i>Z</i>	0
<i>HL</i>	00	00		
<i>PSW</i>	00	00	<i>AC</i>	0
<i>PC</i>	42	0A	<i>P</i>	1
<i>SP</i>	FF	FF		
<i>Int-Reg</i>	00		<i>C</i>	0

For subtraction of two 8-bit numbers by storing address:

; <Program title>

jmp start

;data

;code

start: nop

LDA 0002H

MOV B,A

LDA 0001H

SUB B

STA 0003H

Hlt

Output:

Registers			Flag	
<i>A</i>	05		<i>S</i>	0
<i>BC</i>	02	00		
<i>DE</i>	00	00	<i>Z</i>	0
<i>HL</i>	00	00		
<i>PSW</i>	00	00	<i>AC</i>	0
<i>PC</i>	42	10	<i>P</i>	1
<i>SP</i>	FF	FF		
<i>Int-Reg</i>	00		<i>C</i>	0

Address (Hex)	Address	Data
0000	0	0
0001	1	7
0002	2	2
0003	3	5
0004	4	0
0005	5	0
0006	6	0
0007	7	0
0008	8	0
0009	9	0
000A	10	0
000B	11	0
000C	12	0
000D	13	0

Subtract two 8-bit numbers

Statement: Subtract the contents of memory location 4001H from the memory location 2000H and place the result in memory location 4002H.

1. **Program** –: Subtract two 8-bit numbers
2. **Sample** problem:
3. (4000H) = 51H
4. (4001H) = 19H
5. **Result** = 51H – 19H = 38H
- 6.
7. **Source** program:
8. LXI H, 4000H : "HL points 4000H"
9. MOV A, M : "Get first operand"
10. INX H : "HL points 4001H"
11. SUB M : "Subtract second operand"
12. INX H : "HL points 4002H"
13. MOV M, A : "Store result at 4002H"
14. HLT : "Terminate program execution"