Programming in 8085

Let's see some simple example to demonstrate the use of some important instructions of 8085.

The memory addresses given in the program are for a particular microprocessor kit. These addresses can be changed to suit the microprocessor kit available in your system.

Store 8-bit data in memory

Program

Store 8-bit data in memory using direct addressing

```
MVI A, 49H     : "Store 49H in the accumulator"
STA 2501H      : "Copy accumulator contents at address 2501H"
HLT            : "Stop"
```

Store 8-bit data in memory using indirect addressing

```
LXI H     : "Load H-L pair with 2501H"
MVI M     : "Store 49H in memory location pointed by H-L register pair (2501H)"
HLT       : "Stop"
```

Add two 8-bit numbers

⇧ SCROLL TO TOP

Example

```
(2501 H) = 99H
(2502 H) = 39H
Result (2503 H) = 99H + 39H = D2H
Since,
   1 0 0 1 1 0 0 1 (99H)
 + 0 0 1 1 1 0 0 1 (39H)
   1 1 0 1 0 0 1 0 (D2H)
```

Program

```
LXI H, 2501H   : "Get address of first number in H-L pair. Now H-L points to 2501H"
MOV A, M       : "Get first operand in accumulator"
INX H          : "Increment content of H-L pair. Now, H-L points 2502H"
ADD M          : "Add first and second operand"
INX H          : "H-L points 4002H"
MOV M, A       : "Store result at 2503H"
HLT            : "Stop"
```

Subtract two 8-bit numbers

Example

⇧ SCROLL TO TOP

```
(2501 H) = 49H
(2502 H) = 32H
Result (2503 H) = 49H - 32H = 17H
```

Program

```
LXI H, 2501H    : "Get address of first number in H-L pair. Now H-L points to 2501H"
MOV A, M     : "Get first operand in accumulator"
INX H       : "Increment content of H-L pair. Now, H-L points 2502H"
SUB M       : "Subtract first to second operand"
INX H       : "H-L points 4002H"
MOV M, A     : "Store result at 2503H"
HLT        : "Stop"
```

Add two 16-bits numbers

Add the 16-bit number in memory locations 2501H and 2502H to the 16-bit number in memory locations 2503H and 2504H. The most significant eight bits of the two numbers to be added are in memory locations 2502H and 4004H. Store the result in memory locations 2505H and 2506H with the most significant byte in memory location 2506H.

Example

⇧ SCROLL TO TOP

```
(2501H) = 15H
(2502H) = 1CH
(2503H) = B7H
(2504H) = 5AH

Result = 1C15 + 5AB7H = 76CCH

(2505H) = CCH
(2506H) = 76H
```

Program

Add two 16-bits number with ADD and ADC instruction

```
LHLD 2501H   : "Get Ist 16-bit number in H-L pair"
XCHG        : "Save Ist 16-bit number in DE"
LHLD 2503H   : "Get 2nd 16-bit number in H-L pair"
MOV A, E     : "Get lower byte of the Ist number"
ADD L       : "Add lower byte of the 2nd number"
MOV L, A     : "Store result in L-register"
MOV A, D     : "Get higher byte of the Ist number"
ADC H       : "Add higher byte of the 2nd number with CARRY"
MOV H, A     : "Store result in H-register"
SHLD 4004H   : "Store 16-bit result in memory locations 2505H and 2506H"
```

⇧ SCROLL TO TOP

Add two 16-bits numbers with DAD instruction

```
LHLD 2501H  : "Get 1st 16-bit number"
XCHG        : "Save 1st 16-bit number in DE"
LHLD 2503H  : "Get 2nd 16-bit number in H-L"
DAD D       : "Add DE and HL"
SHLD 2505H  : "Store 16-bit result in memory locations 2505H and 2506H".
HLT         : "Stop"
```

Subtract two 16-bit numbers

Example

```
(2500H) = 19H
(2501H) = 6AH
(2504H) = 15H (2503H) = 5CH
Result    = 6A19H ? 5C15H = OE04H
(2504H) = 04H
(2505H) = OEH
```

Program

```
LHLD 2500H     : "Get first 16-bit number in HL"
XCHG           : "Save first 16-bit number in DE"
LHLD 2502H     : "Get second 16-bit number in HL"
⇧ SCROLL TO TOP  t lower byte of the first number"
```

```
SUB L          : "Subtract lower byte of the second number"
MOV L, A       : "Store the result in L register"
MOV A, D       : "Get higher byte of the first number"
SBB H          : "Subtract higher byte of second number with borrow"
MOV H, A       : "Store l6-bit result in memory locations 2504H and 2505H"
SHLD 2504H     : "Store l6-bit result in memory locations 2504H and 2505H"
HLT            : "Terminate program execution"
```

Add contents of two memory locations

Example

```
(2500H) = 7FH
(2501H) = 89H
Result    = 7FH + 89H = l08H
(2502H) = 08H
(2503H) = 01H
```

Program

```
LXI H, 2500H   : "HL Points 2500H"
MOV A, M       : "Get first operand"
INX H          : "HL Points 250lH"
ADD M          : "Add second operand"
INX H          : "HL Points 2502H"
               re the lower byte of result at 2502H"
```

⇧ SCROLL TO TOP

```
MVIA, 00       : "Initialize higher byte result with 00H"
ADC A          : "Add carry in the high byte result"
INX H          : "HL Points 2503H"
MOV M, A       : "Store the higher byte of result at 2503H"
HLT            : "Terminate program execution"
```

Finding 1's complement of a number

To obtain one's complement of a number its 0 bits are replaced by 1 and 1 by 0.

Example 1

```
(2501H) = 96 H = 1001 0110
                        (9)    (6)
One's complement = 0110 1001 = 69 H
Result = (2502H) = 69H
```

Example 2

```
(2501H) = E4H
Result = (2502H) = 1BH
```

Program

The number is placed in the memory location 2501 H.

The result is stored in the memory location 2502 H.

⇧ SCROLL TO TOP

```
LDA 2501H  : "Get the number IN accumulator"
CMA        : "take its complement"
STA 2502H  : "Store result in 2502H"
HLT        : "Stop"
```

Finding 2's complement of a number

2's complement of a number is obtained by adding 1 to the 1's complement of the number.

Example

```
To find the two's complement of 96.
96 = 1001 0110
1's complement = 0110  1001 = 69
                 +     0000  0001
2's complement = 0110  1010 = 6A
```

Program

The number is placed in the memory location 2501 H.

The result is to be stored in the memory location 2502 H.

```
LDA 2501 H  : "Get data in accumulator"
CMA         : "Take its 1's complement"
              e in the number"
              re the result in 2502 H"
```

⇧ SCROLL TO TOP

```
HLT        : "Stop"
```

Count number of 1's in a number

Example

```
2501 H = 04
2502 H = 34 H
2503 H = A9H
2504 H = 78H
2505 H = 56H
Result = 2503 H = A9H
```

Program

Count number of 1's of the content of the register D and store the count in the register B.

```
MVI B, 00H
MVI C, 08H
MOV A, D
BACK: RAR
JNC SKIP
INR B
SKIP: DCR C
```

⇧ SCROLL TO TOP

Find larger of two numbers

Example

```
2501H = 98 H
2502H = 87H
Result = 98H (2503H)
```

Program

The first number 98H is placed in the memory location 2501 H.

The second number 87H is placed in the memory location 2502H.

The result is stored in the memory location 2503 H.

```
LXI H, 2501H : "Address of first number in H-L pair"
MOV A, M         : "1stt number in accumulator"
INX H        : "Address of 2nd number in H-L pair"
CMP M        : "compare 2nd number with 1st number"
JNC AHEAD        : "No, larger is in accumulator. Go to AHEAD"
MOV A, M         : "Yes, get 2nd number in the accumulator"
STA 2503 H       : "Store larger number in 2503H"
HLT              : "Stop"
```

Find smaller of two numbers

⇧ SCROLL TO TOP

```
2501H = 84 H
2502H = 99 H
Result = 84 H(2503H)
```

Program

The first number 84H is placed in the memory location 2501 H.

The second number 99H is placed in the memory location 2502H.

The result is stored in the memory location 2503 H.

```
LXI H, 2501H : "Address of first number in H-L pair"
MOV A, M         : "1stt number in accumulator"
INX H        : "Address of 2nd number in H-L pair"
CMP M        : "compare 2nd number with 1st number"
JC AHEAD         : "Yes, smaller number is in accumulator. Go to AHEAD"
MOV A, M         : "No, get 2nd number in the accumulator"
STA 2503 H       : "Store smaller number in 2503H"
HLT              : "Stop"
```

Calculate the sum of series of even numbers

Example

```
2500 H = 4H
```

⇧ SCROLL TO TOP

```
2503 H = 13H
2504 H = 22H
Result = 2505 H = 20+22= 42H
```

Program

The numbers are placed in the memory locations 2501 to 2504H.

The sum is to be stored in the memory location 2450H.

As there are 4 numbers in the series, count = 04

The initial value of the sum is made 00. The even number of the series are taken one by one and added to the sum.

```
LDA 2500H
MOV C, A          : "Initialize counter"
MVI B, 00H        : "sum = 0"
LXI H, 2501H      : "Initialize pointer"
BACK: MOV A, M    : "Get the number"
ANI 01H           : "Mask Bit l to Bit7"
JNZ SKIP          : "Don't add if number is ODD"
MOV A, B          : "Get the sum"
ADD M             : "SUM = SUM + data"
MOV B, A          : "Store result in B register"
SKIP: INX H       : "increment pointer"
⇧ SCROLL TO TOP    crement counter"
```

```
JNZ BACK          : "if counter 0 repeat"
STA 2505H          : "store sum"
HLT              : "Stop"
```

Calculate the sum of series of odd numbers

Example

```
2500 H = 4H
2501 H = 9AH
2502 H = 52H
2503 H = 89H
2504 H = 3FH
Result = 2505 H = 89H + 3FH= C8H
```

Program

The numbers are placed in the memory locations 2501 to 2504H.

The sum is to be stored in the memory location 2450H.

As there are 4 numbers in the series, count = 04

The initial value of the sum is made 00. The odd number of the series are taken one by one and added to the sum.

```
LDA 2500H
⇧ SCROLL TO TOP  ɔitialize counter"
```

```
LXI H, 2501H       : "Initialize pointer"
MVI E, 00          : "Sum low = 0"
MOV D, E           : "Sum high = 0"
BACK: MOV A, M     : "Get the number"
ANI 01H            : "Mask Bit 1 to Bit-7"
JZ SKIP            : "Don't add if number is even"
MOV A, E           : "Get the lower byte of sum"
ADD M              : "Sum = sum + data"
MOV E, A           : "Store result in E register"
JNC SKIP
INR D              : "Add carry to MSB of SUM"
SKIP: INX H        : "Increment pointer"
```

Find the square of given number

Program

Find the square of 07 (decimal) using lookup table techniques.

The number 07 D is in the memory.

The result is to be stored in the memory location 2501H.

The table for square is stored from 2600 to 2609 H.

```
LDA 2500H   : "Get data in accumulator"
⇧ SCROLL TO TOP   ata in register L"
```

```
MVI H, 26H  : "Get 26 in register H"
MOV A, M    : "Square of data in accumulator"
STA 2501 H  : "Store Square in 2501 H"
HLT : "Stop"
```

Separate even numbers from given numbers

Program

Let's see the program to separate even numbers from the given list of 50 numbers and store them in other location starting from 2600H.

Starting location of 50 number list is 2500H.

```
LXI H, 2500H       : "Initialize memory pointer 1"
LXI D, 2600H       : "Initialize memory pointer2"
MVI C, 32H         : "Initialize counter"
BACK:MOV A, M      : "Get the number"
ANI 01H            : "Check for even number"
JNZ SKIP           : "If ODD, don't store"
MOV A, M           : "Get the number"
STAX D             : "Store the number in result list"
INX D              : "Increment pointer 2"
SKIP: INX H        : "Increment pointer 1"
DCR C              : "Decrement counter"
⇧ SCROLL TO TOP  f not zero, repeat"
```

```
HLT                 : "Stop"
```

← Prev Next →

Youtube For Videos Join Our Youtube Channel: Join Now

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share

Learn Latest Tutorials

Splunk tutorial Splunk	SPSS tutorial SPSS	Swagger tutorial Swagger

⇧ SCROLL TO TOP

T-SQL tutorial

Transact-SQL

Tumblr tutorial

Tumblr

React tutorial

ReactJS

Regex tutorial

Regex

Reinforcement learning tutorial

Reinforcement Learning

R Programming tutorial

R Programming

RxJS tutorial

RxJS

React Native tutorial

React Native

Python Design Patterns

Python Design Patterns

Python Pillow tutorial

Python Pillow

Python Turtle tutorial

Python Turtle

Keras tutorial

Keras

Preparation

Aptitude

⇧ SCROLL TO TOP

Logical Reasoning

Verbal Ability

Verbal Ability

Reasoning

Interview Questions

Interview Questions

Company Interview Questions

Company Questions

Trending Technologies

Artificial Intelligence Tutorial

Artificial Intelligence

AWS Tutorial

AWS

Selenium tutorial

Selenium

Cloud Computing tutorial

Cloud Computing

Hadoop tutorial

Hadoop

ReactJS Tutorial

ReactJS

Data Science Tutorial

Data Science

Angular 7 Tutorial

Angular 7

Blockchain Tutorial

Blockchain

⇧ SCROLL TO TOP

Git Tutorial
Git

Machine Learning Tutorial
Machine Learning

DevOps Tutorial
DevOps

B.Tech / MCA

DBMS tutorial
DBMS

Data Structures tutorial
Data Structures

DAA tutorial
DAA

Operating System tutorial
Operating System

Computer Network tutorial
Computer Network

Compiler Design tutorial
Compiler Design

Computer Organization and Architecture
Computer Organization

Discrete Mathematics Tutorial
Discrete Mathematics

Ethical Hacking Tutorial
Ethical Hacking

Computer

⇧ SCROLL TO TOP

Software Engineering Tutorial

html tutorial
Web Technology

Software Engineering

Cyber Security tutorial

Cyber Security

Automata Tutorial

Automata

C Language tutorial

C Programming

C++ tutorial

C++

Java tutorial

Java

.Net Framework tutorial

.Net

Python tutorial

Python

List of Programs

Programs

Control Systems tutorial

Control System

Data Mining Tutorial

Data Mining

Data Warehouse Tutorial

Data Warehouse

⇧ SCROLL TO TOP