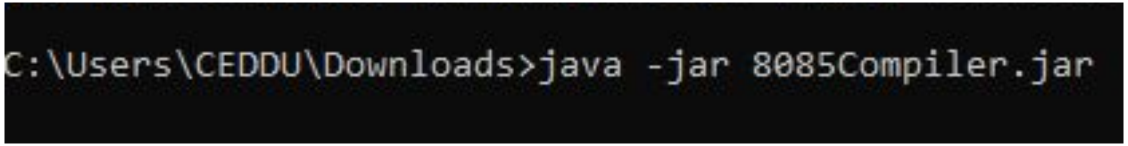
Computer Organization and Architecture 8085 Instructions Set

Prepared By: Prof. Preeti Sharma
Assistant Professor, CE, Faculty of Technology, Dharmsinh Desai University, Nadiad.
Ph.D. Pursuing CE/IT, LDCE, GTU, Ahmedabad
M.E. IT, LDCE, Ahmedabad
B.E. CE, VGEC, Chandkheda, Ahmedabad
D.E. CE, GPG, Ahmedabad

8085 Instruction Set:

1. Data Transfer Instructions
2. Arithmetic Instructions
3. Logical Instructions
4. Branching Instructions
5. Control Instructions

Simulator: https://learning-microprocessors.sourceforge.io/8085-simulator.html
Starting the Simulator:



```
C:\Users\CEDDU\Downloads>java -jar 8085Compiler.jar
```

Data Transfer Instructions

Data Transfer Instructions:

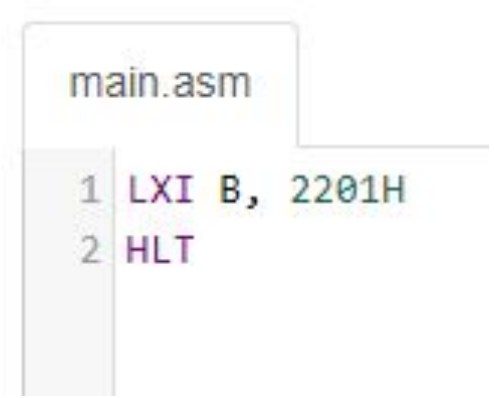
1. MOV
2. MVI
3. LDA
4. STA
5. LXI
6. LDAX
7. STAX
8. LHLD
9. SHLD
10. XCHG
11. XTHL
12. PUSH
13. POP
14. IN
15. OUT

LXI Instruction:

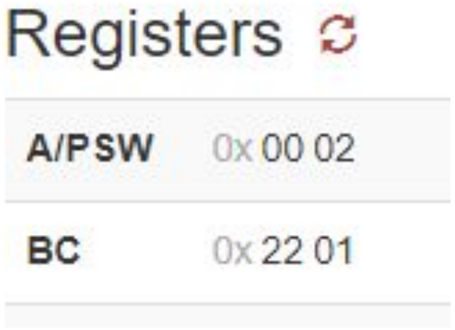
LXI (Load register pair immediate): - This instruction loads a 16-bit immediate value into the specified register pair (e.g., HL).

Example: - LXI B, 2201H (2201H is stored in BC pair so that it act as memory pointer)

\

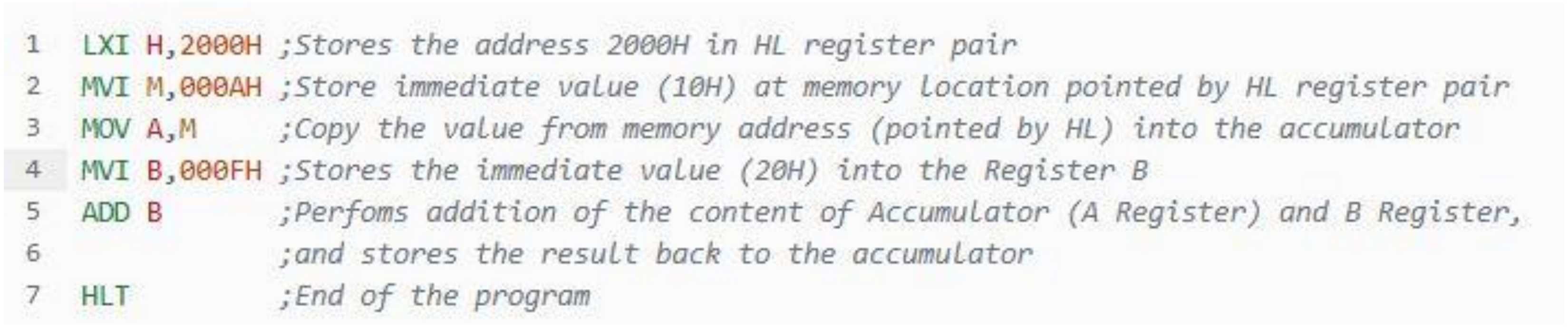

main.asm

```
LXI B, 2201H
HLT
```

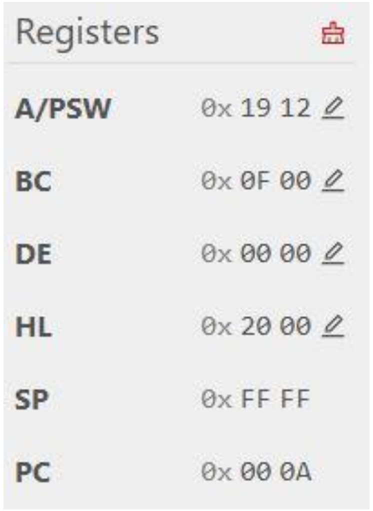

Registers

A/PSW	0x 00 02
BC	0x 22 01

EXAMPLE:



```
LXI H,2000H ;Stores the address 2000H in HL register pair
MVI M,000AH ;Store immediate value (10H) at memory location pointed by HL register pair
MOV A,M     ;Copy the value from memory address (pointed by HL) into the accumulator
MVI B,000FH ;Stores the immediate value (20H) into the Register B
ADD B       ;Perfoms addition of the content of Accumulator (A Register) and B Register,
            ;and stores the result back to the accumulator
HLT         ;End of the program
```

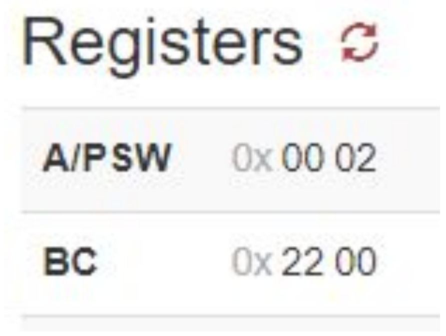


Registers

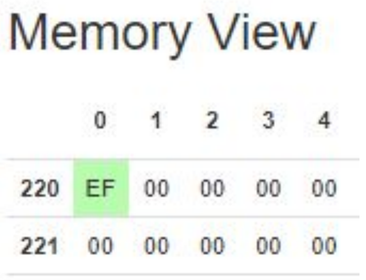
A/PSW	0x 19 12
BC	0x 0F 00
DE	0x 00 00
HL	0x 20 00
SP	0x FF FF
PC	0x 00 0A

LDAX Instruction:

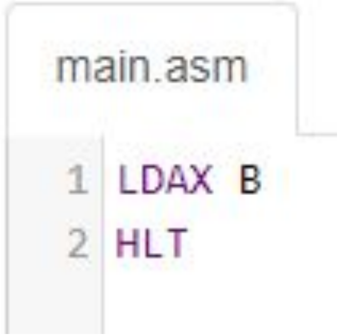
LDAX: In 8085 Instruction set, **LDAX** is a mnemonic that stands for **LoaD Accumulator** from memory pointed by **eXtended register pair** denoted as "rp" in the instruction.
This instruction loads the accumulator (A) with the data from the memory location pointed to by the specified register pair (e.g., HL or DE).
This instruction uses register indirect addressing mode for specifying the data. It occupies only 1-Byte in the memory.
Example: LDAX B or LDAX **B**
Let us suppose that the initial content of BC register pair is 2200H and at memory location 2200H is having the 8-bit content of EFH. So after execution of the instruction the Accumulator will have the updated content of EFH.


Registers

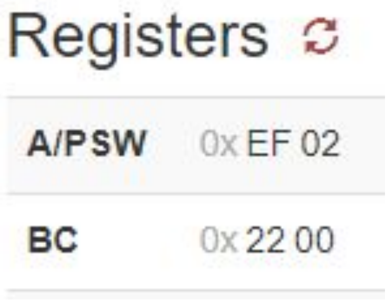
A/PSW	0x 00 02
BC	0x 22 00


Memory View

	0	1	2	3	4
220	EF	00	00	00	00
221	00	00	00	00	00


main.asm

```
LDAX  B
HLT
```


Registers

A/PSW	0x EF 02
BC	0x 22 00

STAX Instruction:

- This instruction stores the content of the accumulator into the memory location pointed to by the specified register pair (e.g., HL or DE).

EXAMPLE: STAX B or STAX D

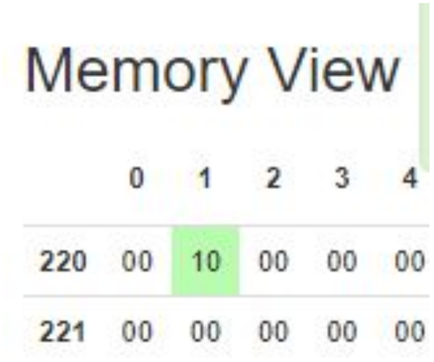
LHLD Instruction:

LHLD (Load H and L register direct): - this instruction loads the contents of the 16- bit memory location into the HL register pair.

EXAMPLE: LHLD 2200H (the content of location 2200H is copied into the HL reg pair)

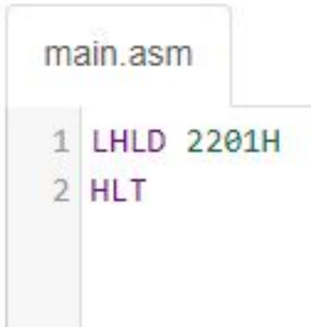
Operation:

- The **lower byte** (byte at the lower address) is loaded into **L**.
- The **higher byte** (byte at the higher address) is loaded into **H**.



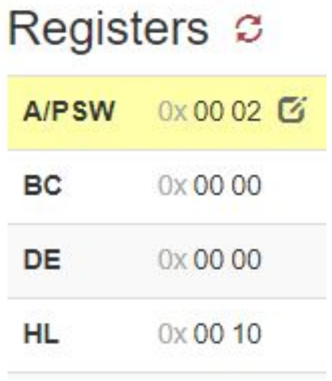
Memory View

	0	1	2	3	4
220	00	10	00	00	00
221	00	00	00	00	00



main.asm

```
LHLD 2201H
HLT
```



Registers

A/PSW	0x 00 02
BC	0x 00 00
DE	0x 00 00
HL	0x 00 10

SHLD Instruction:

SHLD stands for **Store HL Direct**.

It stores the contents of the **HL register pair** into two memory locations:

- The **lower byte** (stored in register L) is stored at the **specified memory address**.
- The **higher byte** (stored in register H) is stored at the **next memory address** (i.e., the memory address is incremented by 1).

Example:

Let's say the HL register pair contains the value 1234H (H = 12H and L = 34H), and you want to store this value starting at memory location 5000H. After the SHLD 5000H instruction, the contents of the memory will be:

- **Memory location 5000H**: 34H (lower byte from L)
- **Memory location 5001H**: 12H (higher byte from H)

XCHG Instruction:

- The **XCHG** instruction in the 8085 microprocessor is used to **exchange the contents** of the **HL** register pair with the **DE** register pair. It swaps the values between registers **H, L** and **D, E**.
- **EXAMPLE: XCHG**

Operation:

- The **contents of the HL register pair** (i.e., H and L) are exchanged with the **contents of the DE register pair** (i.e., D and E).
  - The value of register H is exchanged with register D.
  - The value of register L is exchanged with register E.

Where it can be used?

Consider implementing a simple sorting algorithm like **bubble sort**. In bubble sort, adjacent elements are compared, and if they are out of order, they are swapped. The **XCHG** instruction can efficiently swap the two values held in registers for this purpose.

Arithmetic Instructions

Arithmetic Instructions:

1. ADD
2. ADI
3. ADC
4. ACI
5. SUB
6. SUI
7. SBB
8. SBI
9. INR
10. INX
11. DCR
12. DCX
13. DAD

DAD Instruction:

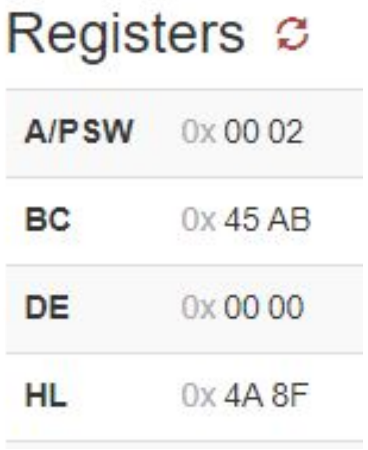
- The **DAD (Double Add)** instruction in the **8085 microprocessor** is used primarily for handling **16-bit additions**.
- Since, **8085** is an 8-bit microprocessor, the **DAD** instruction provides an efficient way to perform **16-bit additions** by adding two 16-bit register pairs in one instruction.
- **DAD: -** Add register pair to HL register**.**
- 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.
- **Syntax: DAD rp**
- **Example: DAD B or DAD D or DAD H**

The **DAD** instruction performs the following operations:

1. It adds the 16-bit value in the specified register pair to the **HL** register pair.
2. The result is stored in the **HL** register pair (i.e., HL = HL + register pair).
3. The **carry flag (CY)** is updated based on the result of the addition. If there is a carry-out (i.e., if the sum exceeds 16 bits), the **CY** flag will be set to 1. Otherwise, it will be reset to 0.

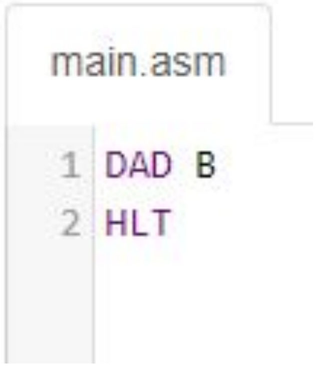
Using “DAD” Double Add instruction:

EXAMPLE – 1: Assembly Program to add 16 bits content specified by register pair with HL register pair.



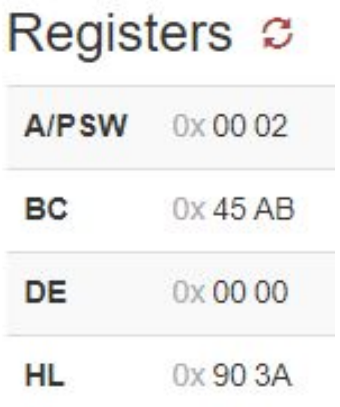
Registers

A/PSW	0x 00 02
BC	0x 45 AB
DE	0x 00 00
HL	0x 4A 8F



main.asm

```
DAD  B
HLT
```



Registers

A/PSW	0x 00 02
BC	0x 45 AB
DE	0x 00 00
HL	0x 90 3A

Logical Instructions

Logical Instructions:

1. ANA (Logical AND with Accumulator)
2. ANI (Logical AND with Immediate)
3. ORA (Logical OR with Accumulator)
4. ORI (Logical OR with Immediate)
5. XRA (Logical XOR with Accumulator)
6. XRI (Logical XOR with Immediate)
7. RLC (Rotate Left Accumulator)
8. RRC (Rotate Right through Carry)
9. RAL (Rotate Accumulator Left through Carry)
10. RAR (Rotate Accumulator Right through Carry)
11. CMA (Complement Accumulator)
12. CMC (Complement Accumulator)
13. CMP (Compare Accumulator)
14. CPI (Compare Accumulator with Immediate)
15. STC (Set Carry)

ANA and ANI Instruction in 8085:

ANA: Performs Logical AND operation is performed with the specified register or memory with accumulator.

Example: **ANA B**

ANA M

ANI: Performs logical AND operation is performed between accumulator and specified immediate data (with Immediate Addressing mode).

Example: ANI 30H

ORA and ORI Instruction in 8085:

- In 8085 Instruction set, **ORA** is a mnemonic, which stands for “OR Accumulator” and “R” stands for any of the following registers, or memory location M pointed by HL pair.

  R = A, B, C, D, E, H, L, or M

- This instruction is used to OR contents of R with the Accumulator. The result of OR operation will be stored back in the Accumulator.
- It occupies only 1-Byte in memory.

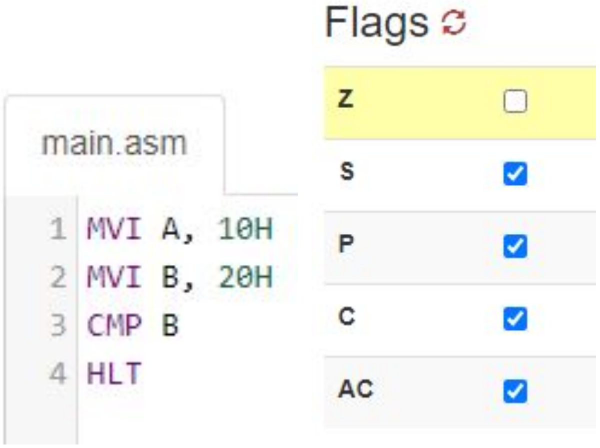
- **ORI: -** Logical OR operation is performed between accumulator and immediate data.
- **EXAMPLE: ORI 30H**

CMP Instruction:

CMP : Compare the content of Register or Memory with the content of 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.

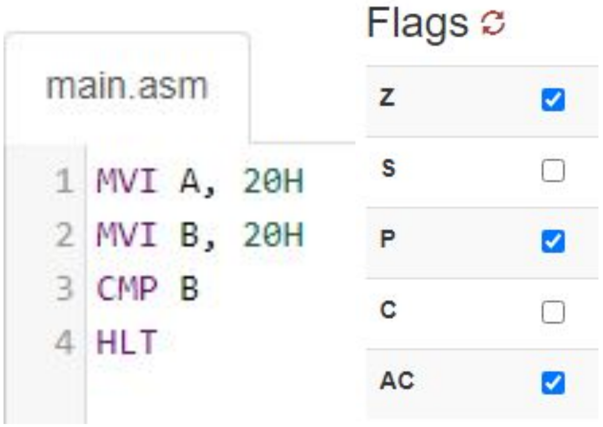


main.asm

```
MVI A, 10H
MVI B, 20H
CMP B
HLT
```

Flags

Z	☐
S	✓
P	✓
C	✓
AC	✓

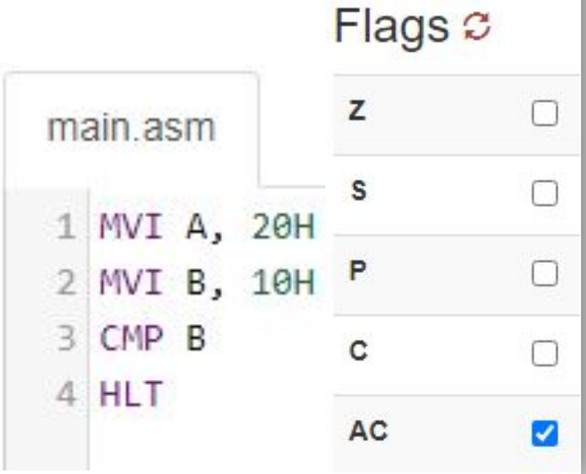


main.asm

```
MVI A, 20H
MVI B, 20H
CMP B
HLT
```

Flags

Z	✓
S	☐
P	✓
C	☐
AC	✓



main.asm

```
MVI A, 20H
MVI B, 10H
CMP B
HLT
```

Flags

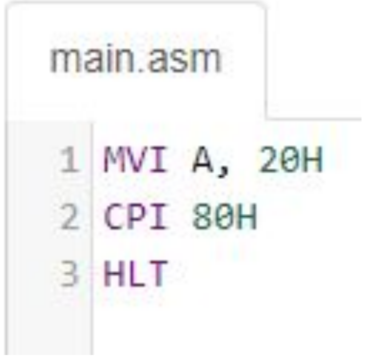
Z	☐
S	☐
P	☐
C	☐
AC	✓

CPI Instruction:

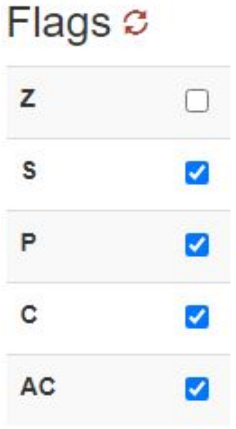
CPI : Compare Immediate with Accumulator.

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


main.asm

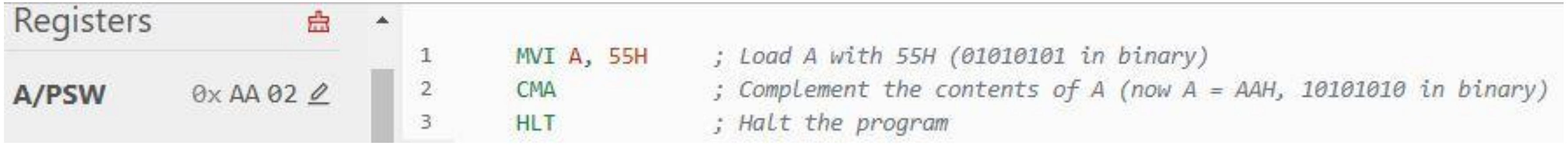
```
MVI A, 20H
CPI 80H
HLT
```


Flags

Z	☐
S	✓
P	✓
C	✓
AC	✓

CMA (Complement Accumulator):

- The CMA (Complement Accumulator) instruction in the 8085 microprocessor is used to **complement (invert) all the bits** in the **accumulator**. Essentially, this means that each bit in the accumulator is changed from **1 to 0** and from **0 to 1**.

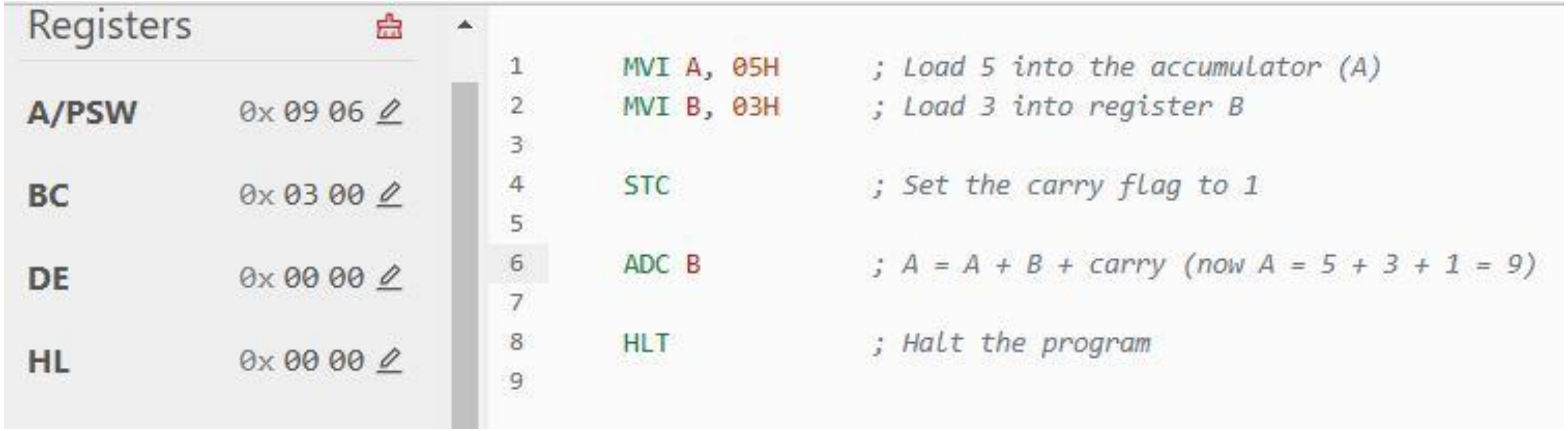


A/PSW	0x AA 02

```
MVI A, 55H      ; Load A with 55H (01010101 in binary)
CMA             ; Complement the contents of A (now A = AAH, 10101010 in binary)
HLT             ; Halt the program
```

STC (Set Carry Flag) Instruction:

- The STC instruction in the 8085 microprocessor stands for **Set Carry Flag**. This instruction is used to **set the carry flag** (CY) to **1** explicitly.



Registers

A/PSW	0x 09 06
BC	0x 03 00
DE	0x 00 00
HL	0x 00 00

```
MVI A, 05H        ; Load 5 into the accumulator (A)
MVI B, 03H        ; Load 3 into register B

STC               ; Set the carry flag to 1

ADC B             ; A = A + B + carry (now A = 5 + 3 + 1 = 9)

HLT               ; Halt the program

```

Thanks