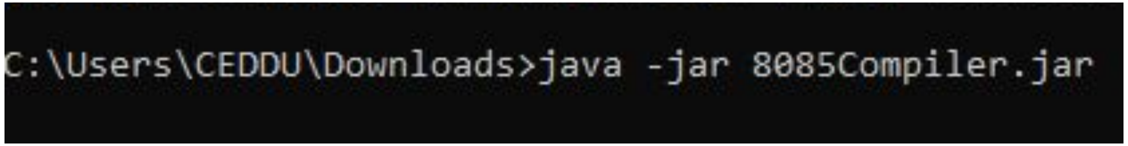
Computer Organization and Architecture 8085 Instructions Set

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
```

Logical Instructions

Logical Instructions:

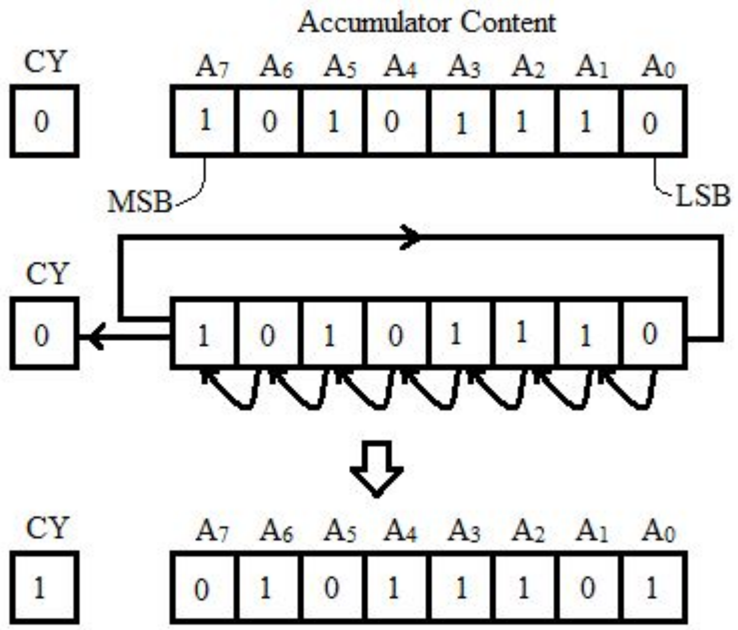
1. ANA (Logical AND with Accumulator)
2. ANI (Logical AND with Immediate)
3. ORA (Logical OR with Accumulator)
4. ORI (Logical OR with Immediate)
5. XRA (Logical XOR with Accumulator)
6. XRI (Logical XOR with Immediate)
7. RLC (Rotate Left Accumulator)
8. RRC (Rotate Right through Carry)
9. RAL (Rotate 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)

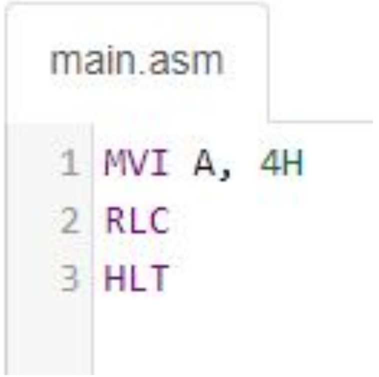
CMC (Complement Carry):

- The CMC (Complement Carry) instruction in the **8085 microprocessor** is used to **complement** (invert) the **carry flag**. If the carry flag is **1**, the CMC instruction will change it to **0**, and if the carry flag is **0**, it will change it to **1**. In simple terms, it toggles the carry flag.

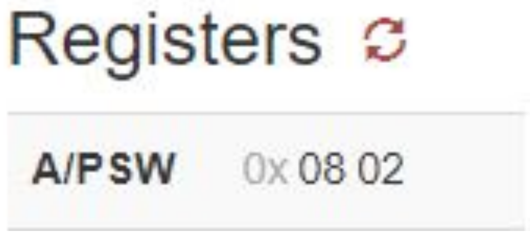
RLC (Rotate Accumulator Left) Instruction:

RLC: Each binary bit of the accumulator is rotated left by one position.




main.asm

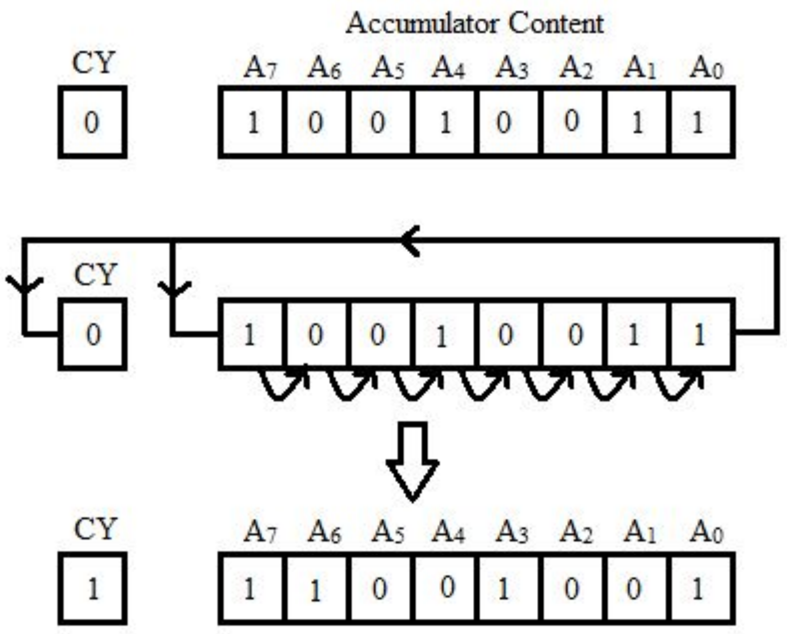
```
MVI A, 4H
RLC
HLT
```


Registers

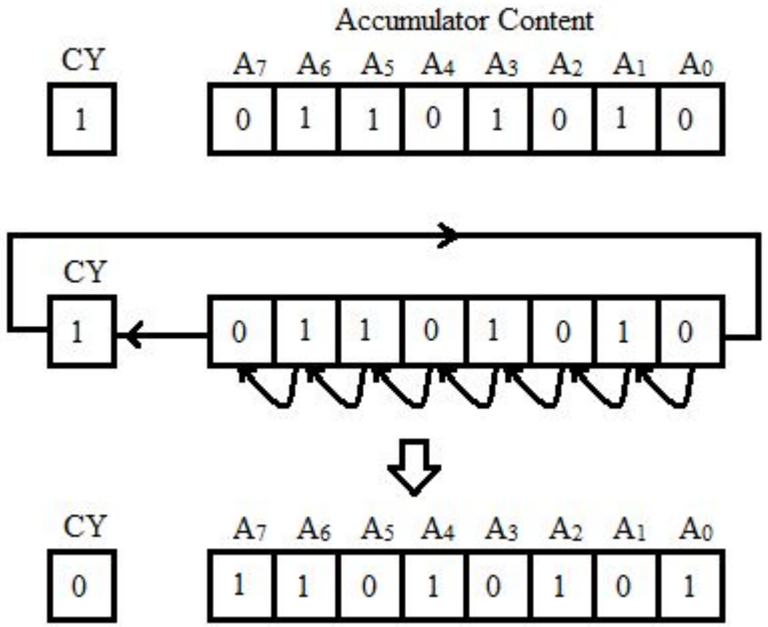
A/PSW	0x 08 02

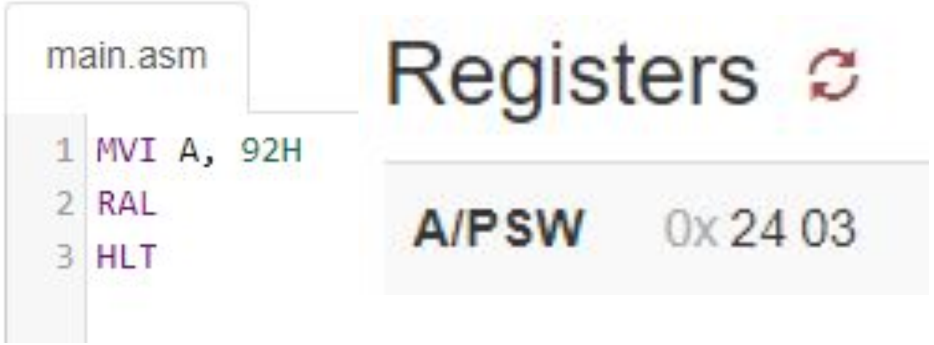
RRC (Rotate Accumulator Right):

- In **RRC**, the bit that is shifted out (from the least significant bit, LSB) is placed into the carry flag, and the carry flag's previous value is placed into the most significant bit (MSB) of the accumulator.



RAL(Rotate the accumulator left through carry):



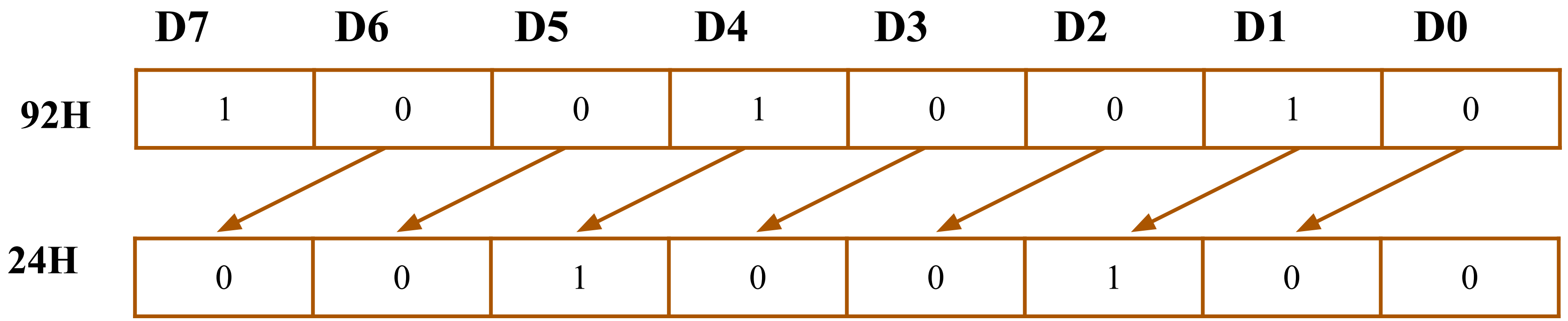

main.asm

```
MVI A, 92H
RAL
HLT
```

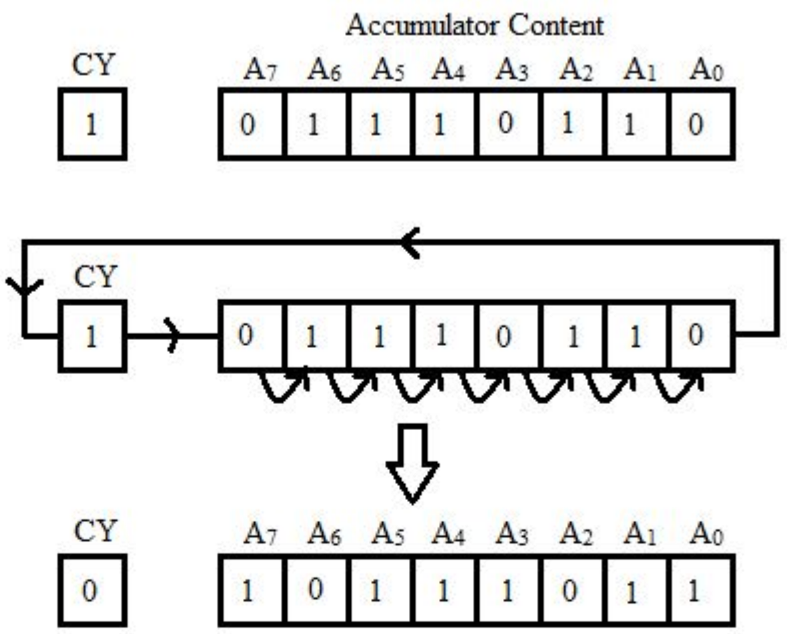
Registers

A/PSW	0x 24 03

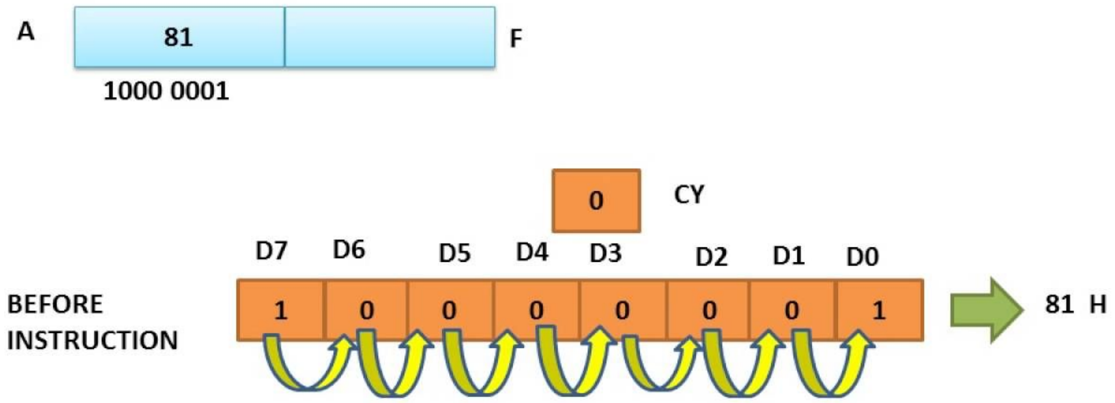
EXAMPLE: Each binary bit of the accumulator is rotated left by one position through the Carry flag. Bit D7 is placed in the Carry flag, and the Carry flag is placed in the least significant position D0. CY is modified according to bit D7.



RAR (Rotate Accumulator Right through Carry):



RAR = rotate accumulator right through carry



Branch Instructions

Branch Instructions:

1. **Unconditional Branch**
   1.1. JMP (Jump without checking any conditions)
2. **Conditional Branch**
   2.1. JC (Jump if Carry)
   2.2. JNC (Jump if not Carry)
   2.3. JZ (Jump if Zero)
   2.4. JNZ (Jump if not Zero)
   2.5. JP (Jump if Positive)
   2.6. JM (Jump if Minus)
   2.7. JPE (Jump if Parity Even)
   2.8. JPO (Jump if Parity Odd)

Unconditional Branch:

JMP : (unconditionally jump) The program sequence is transferred to the memory location specified by the 16-bit address given in the operand.

Eg: - JMP 2034H (jump to location 2034H) there is no condition to jump.
JMP ABC (jump to ABC level)

Conditional Branch:

- Conditional branching in the 8085 microprocessor allows the program to jump to a new location based on the value of a flag bit.
- All conditional jump instructions in the 8085 are 3-byte instructions.
- **Conditional jump instructions**: Transfer the program sequence to a memory location based on the following conditions:
  - **JC (Jump if Carry)**: If the result generates carry and CY = 1
  - **JNC (Jump if No Carry)**: If CY = 0
  - **JZ (Jump if Zero)**: If the result is zero and Z = 1
  - **JNZ (Jump if not Zero)**: **Z = 0**
  - **JP (Jump if Positive): S=0**
  - **JM (Jump if Minus): S=1**
  - **JPE (Jump if Parity Even): P=1**
  - **JPO (Jump if Parity Odd): P=0**

JC and JNC Instruction:

JC: The program sequence is transferred to a particular level or a 16-bit address if C=1 (or carry is 1)

Eg: - JC ABC (jump to the level abc if C=1)

JNC: The program sequence is transferred to a particular level or a 16-bit address if C=0 (or carry is 0)
Eg: JNC ABC (jump to the level abc if C=0)

JZ and JNZ Instruction:

JZ: The program sequence is transferred to a particular level or a 16-bit address if Z=1 (or zero flag is 0)
Eg: - JZ ABC (jump to the level abc if Z=1)

JNZ: The program sequence is transferred to a particular level or a 16-bit address if Z=0 (or zero flag is 0)
Eg: - JNZ ABC (jump to the level abc if Z=0)

JP and JM Instruction:

JP: The program sequence is transferred to a particular level or a 16-bit address if S=0 (or sign is 0)
Syntax: JP Label or JP Memory operand
Eg: - JP ABC (jump to the level abc if S=0) Will affect the sign flag
JP 2050 ;Jumps to the address if sign flag is 0

JM: The program sequence is transferred to a particular level or a 16-bit address if S=1 (or sign is 1)
Eg: - JM ABC (jump to the level abc if S=1)
JM 2050 ; Jumps to the address if sign flag is 1

JPE and JPO Instruction:

JPE: The program sequence is transferred to a particular level or a 16-bit address if P=1 (or parity flag is 1)
Eg: - JPE ABC (jump to the level abc if P=1)

JPO: The program sequence is transferred to a particular level or a 16-bit address if P=0 (or parity flag is 0)
Eg: - JPO ABC (jump to the level abc if P=0)

Thanks