Microprocessor
Lecture 4

8085 Microprocessor

Presented by: Robinhood Khadka

Instruction Group

- Functionally, the instruction group can be classified into 5 groups

1. Data transfer Group
2. Arithmetic Group
3. Logical Group
4. Branching Group
5. Machine(Miscellaneous Group)

1. Data Transfer Group

- This group of instructions provides the 8085 with the ability to move data around inside the RA, between the RAM and the register of the microprocessor
- 16 instructions are in this group to carry data from source to destination

❑ MOV (Move) :Copies the content of source register Rs into destination register Rd

MOV Rd, Rs

The content of source register is not changed

e.g. MOV A, B

MOV B,C

MOV C,M

If one of the operand is a memory location, it is specified by the content of the register pair HL

1. Data Transfer Group

- ❑ MVI- Move immediate 8-bit data
  - The 8-bit data are stored in the accumulator register or other registers
  - If the operand is a memory location, it is specified by HL register pair
  - i.e. MVI A, 8-bit data
  - MVI B,03
  - MVI D, AB
  - MVI M, data (Memory location pointed by HL pair)

Arithmetic Instructions

- DAD

The 16-bit content of the specified register pair are added to the contents of HL register pair and the result is placed in HL register pair

e.g. DAD Rp

Let H=25 and L=54

and D=26 , E=23

DAD D will function as $[HL] \longleftarrow [HL] + [DE]$

Arithmetic Instructions

- DAA(Decimal Adjust Accumulator)
  - The contents of accumulator are changed form binary value to the 4bit BCD digits
  - If the value of low order four bit (D3-D0) in the accumulator is greater than 9 on AC, Flag is set then 6 is added to the lower 4bit
  - Same procedure is followed for higher 4 bit

For example

```
MVI B , 12H
MVI A , 39H
ADD B
DAA
HLT
```

Arithmetic Instructions

- 12-----0001 0010

  39-----0011 1001

  0100 1011---4B

Where 0100 is 4 < 9 and 1011 is B >9. Therefore we add 6

0100 1011

0000 0110

0101 0001----51

Logical Instructions

1. ANA (Logical AND with Accumulator)

   The content of the operand are logically ANDed with the contents of the accumulator and the result is stored in the accumulator

   i.e. ANA Reg

   ANA M

2. ANI (AND Immediate)

   The 8 bit data are logically ANDed with acc and the result is placed in acc

   i.e. ANI 97H

Logical Instructions

- ORA (Logical OR with Accumulator)

  The content of the acc is logically ORed with acc and the result is placed in the acc

  i.e. ORA Reg

  ORA M

- ORI (Logically OR Immediate)

  The 8 bit data is logically ORed with acc and the result is placed in the acc.

  i.e. ORI 25 H

Logical Instructions

- XRA (Exclusive OR with Accumulator)
  - The contents of the register or memory is logically XORed with acc and the result is placed in the acc

    i.e. XRA Reg

    XRA M

- XRI (Exclusive OR Immediate)

  The 8 bit data is logically XORed with acc and the result is placed in the acc.

  i.e. XRI 01 H

NOTE: In ORA , ORI, XRA, XRI

The Carry Flag is Reset and Auxiliary Carry is Set

Rotate Accumulator

- RLC: Rotate Accumulator Left without carry
  - Each binary bit of the accumulator is rotated left by one position
  - Bit D7 is placed in the position D0 as well as the carry flags
  - Here carry flag is modified according to bit D7
  - RLC (No operand)

Rotate Accumulator

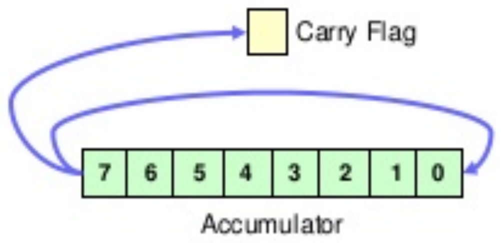
- RAL: Rotate Accumulator left with carry
  - Each binary bit of the accumulator is rotated by 1 position with carry flags
  - Bit D7 is placed in the carry flag
  - The carry flag is placed in the LSB position DO

RAL (No Operand)

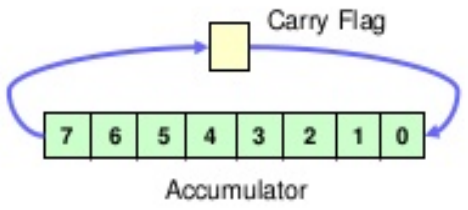
RLC and RAL

RLC vs. RAL

- RLC



- RAL



39

Rotate Accumulator

- RRC (Rotate accumulator right without carry)
- RAR (Rotate accumulator right with carry)

Logical instructions

- CMA : Complement Accumulator
  - The contents of the accumulator are complemented
  - CMA (No Operand)
- CMC : Complement Carry
  - Only the flag is complemented
  - CMC(No operand)

Branching Operation

- The branching operation allow the microprocessor to change the sequence of the program either conditionally or unconditionally

- They are categorized in 3 ways
  - Jump Instruction
  - Call and Return Instruction(subroutine)
  - Restart Instruction(Interrupt)

Jump Instruction

- It allows the program sequence to jump to a new memory
- Two types of JUMP's are
  - Unconditional jump
  - Conditional jump

a. JMP: Jump Unconditionally

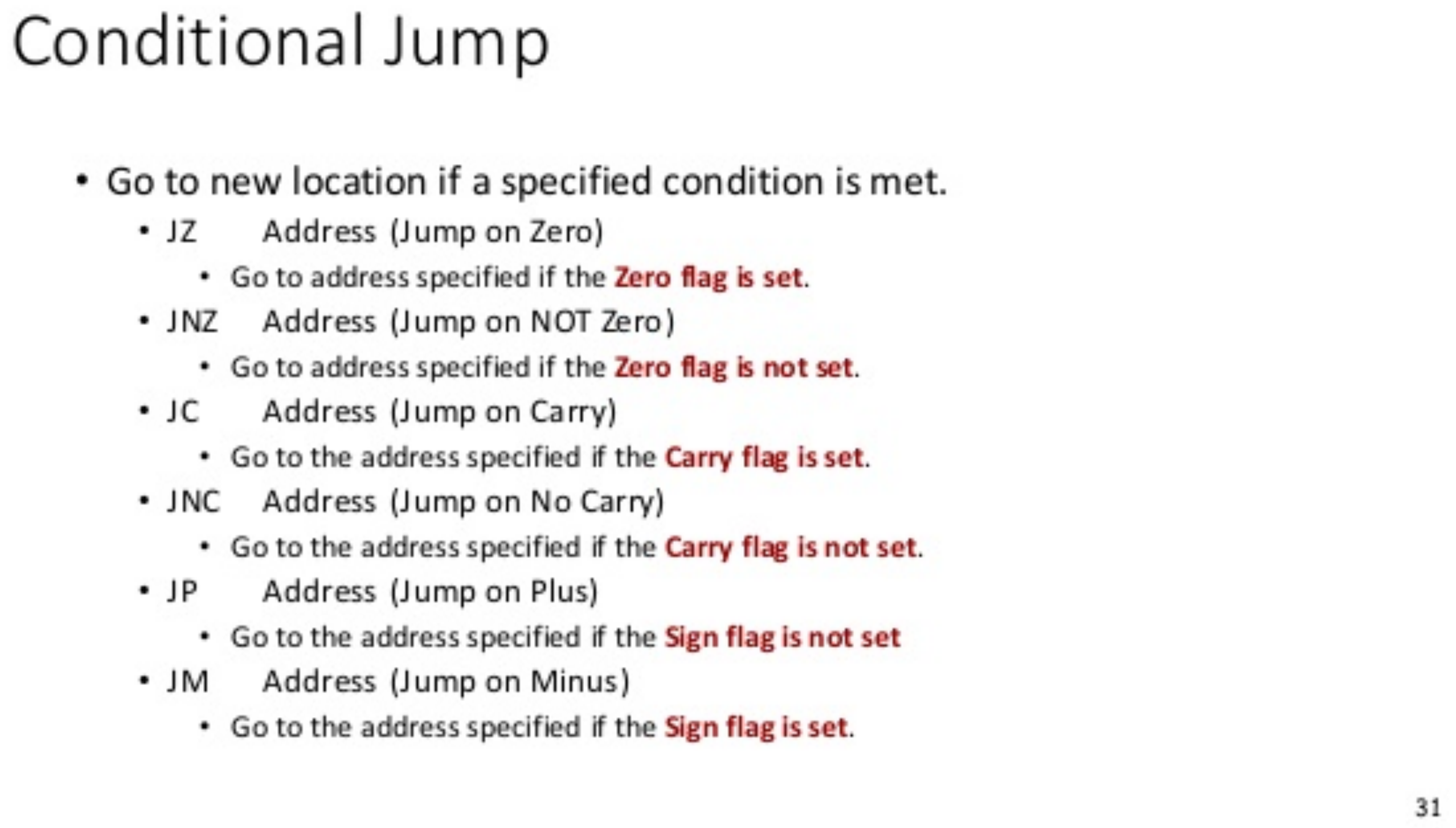
- It simply loads the specified address into the program counter register

-enables programmer to setup load without considering any conditions

i.e. JMP 16-bit Address

e.g. JMP 4500H set the address of PC to 4500H

Jump Instruction



Conditional Jump

- Go to new location if a specified condition is met.
  - JZ Address (Jump on Zero)
    - Go to address specified if the **Zero flag is set**.
  - JNZ Address (Jump on NOT Zero)
    - Go to address specified if the **Zero flag is not set**.
  - JC Address (Jump on Carry)
    - Go to the address specified if the **Carry flag is set**.
  - JNC Address (Jump on No Carry)
    - Go to the address specified if the **Carry flag is not set**.
  - JP Address (Jump on Plus)
    - Go to the address specified if the **Sign flag is not set**
  - JM Address (Jump on Minus)
    - Go to the address specified if the **Sign flag is set**.

31

Programs

- Find the 1’s complement of the number stored at memory location C050H and store the result in register E
- WAP to find the 2’s complement of the number stored at memory location C055H and store the result in D055H
- WAP to subtract two number , 05 as immediate and other number from memory location [D050]-02 and store the result in any memory location
- WAP to multiply 05 and 03 . Store the result in memory location FFFF.