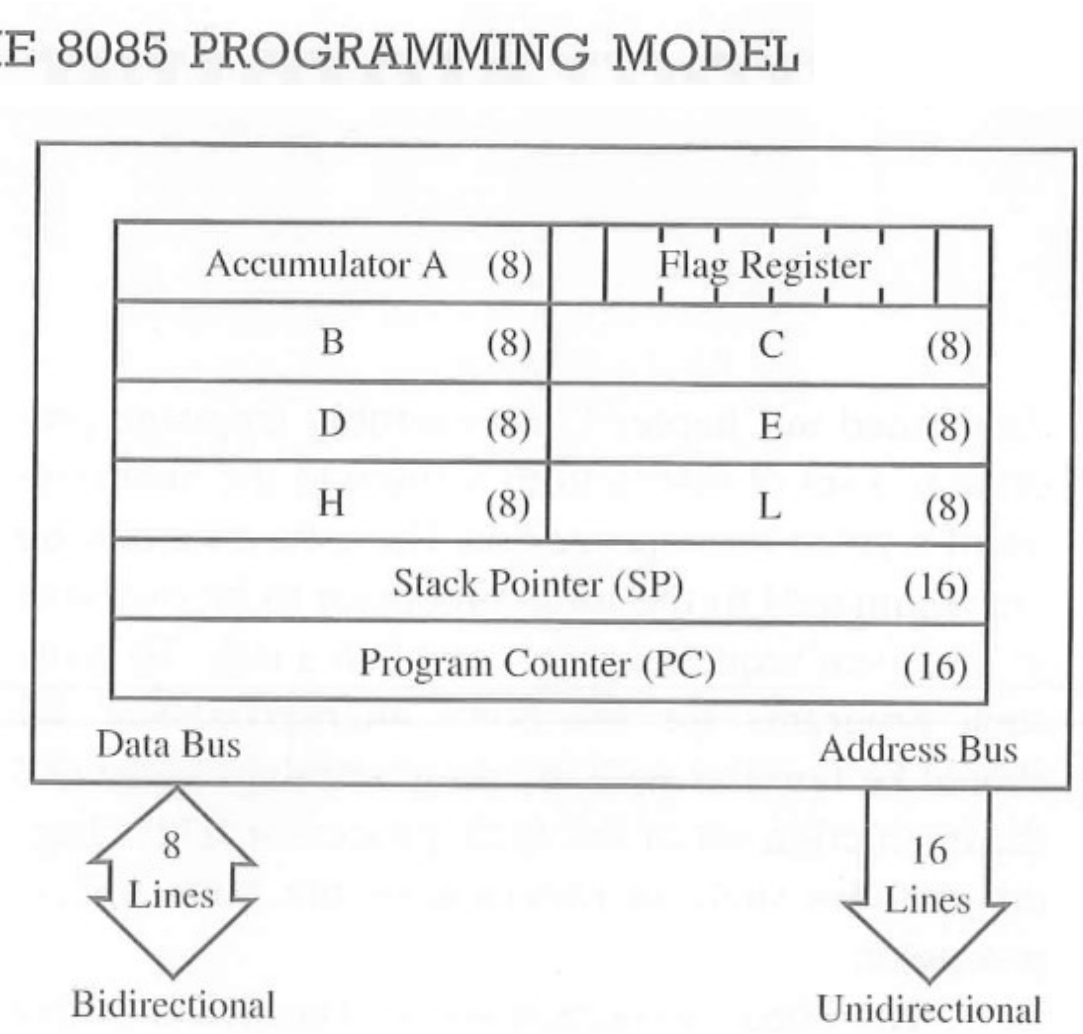
# Introduction to 8085 Assembly language

The Programming Model gives the information required to write programs



# The 8085 Instruction Set

An **instruction** is a binary pattern designed inside a microprocessor to perform a specific function. The entire group of instructions, called the **instruction set,** determines what functions the microprocessor can perform.

The 8085 instructions can be classified into the following five functional categories: data transfer (copy) operations, arithmetic operations, logical operations, branching operations, and machine-control operations.

## DATA TRANSFER (COPY) OPERATIONS

This group of instructions copies data from a location called a source to another location, called a destination, without modifying the contents of the source. In technical manuals, the term *data transfer* is used for this copying function. However, the term *transfer* is misleading; it creates the impression that the contents of a source are destroyed when, in fact, the contents are retained without any modification. The various types of data transfer (copy) are listed below together with examples of each type:

| Types | Examples |
|---|---|
| ☐ Between registers | Copy the contents of register B into register D. |
| ☐ Specific data byte to a register or a memory location | Load register B with the data byte 32H. |
| ☐ Between a memory location and a register | From the memory location 2000H to register B. |
| ☐ Between an I/O device and the accumulator | From an input keyboard to the accumulator. |

## ARITHMETIC OPERATIONS

These instructions perform arithmetic operations such as addition, subtraction, increment, and decrement.

- **Addition**—Any 8-bit number, or the contents of a register, or the contents of a memory location can be added to the contents of the accumulator and the sum is stored in the accumulator. No two other 8-bit registers can be added directly (e.g., the contents of register B cannot be added directly to the contents of register C). The instruction DAD is an exception; it adds 16-bit data directly in register pairs.
- **Subtraction**—Any 8-bit number, or the contents of a register, or the contents of a memory location can be subtracted from the contents of the accumulator and the results stored in the accumulator. The subtraction is performed in 2's complement, and the results, if negative, are expressed in 2's complement. No two other registers can be subtracted directly.
- **Increment/Decrement**—The 8-bit contents of a register or a memory location can be incremented or decremented by 1. Similarly, the 16-bit contents of a register pair (such as BC) can be incremented or decremented by 1. These increment and decrement operations differ from addition and subtraction in an important way; i.e., they can be performed in any one of the registers or in a memory location.

## LOGICAL OPERATIONS

These instructions perform various logical operations with the contents of the accumulator.

- **AND, OR, Exclusive-OR**—Any 8-bit number, or the contents of a register, or of a memory location can be logically ANDed, ORed, or Exclusive-ORed with the contents of the accumulator. The results are stored in the accumulator.
- **Rotate**—Each bit in the accumulator can be shifted either left or right to the next position.
- **Compare**—Any 8-bit number, or the contents of a register, or a memory location can be compared for equality, greater than, or less than, with the contents of the accumulator.
- **Complement**—The contents of the accumulator can be complemented; all 0s are replaced by 1s and all 1s are replaced by 0s.

## BRANCHING OPERATIONS

This group of instructions alters the sequence of program execution either conditionally or unconditionally.

- **Jump**—Conditional jumps are an important aspect of the decision-making process in programming. These instructions test for a certain condition (e.g., Zero or Carry flag) and alter the program sequence when the condition is met. In addition, the instruction set includes an instruction called *unconditional jump.*
- **Call, Return, and Restart**—These instructions change the sequence of a program either by calling a subroutine or returning from a subroutine. The conditional Call and Return instructions also can test condition flags.

## MACHINE CONTROL OPERATIONS

These instructions control machine functions such as Halt, Interrupt, or do nothing.

# INSTRUCTION

An **instruction** is a command to the microprocessor to perform a given task on specified data. Each instruction has two parts: one is the task to be performed, called the **operation code** (op-code), and the second is the data to be operated on, called the **operand.** The operand (or data) can be specified in various ways. It may include 8-bit (or 16-bit) data, an internal register, a memory location, or an 8-bit (or 16-bit) address. In some instructions, the operand is implicit.

# Instruction Word Size

The 8085 instruction set is classified into the following three groups according to word size or byte size.

1. 1-byte instructions
2. 2-byte instructions
3. 3-byte instructions

## ONE-BYTE INSTRUCTIONS

A 1-byte instruction includes the opcode and the operand in the same byte. For example:

| Task | Opcode | Operand* | Binary Code | Hex Code |
|---|---|---|---|---|
| Copy the contents of the accumulator in register C. | MOV | C,A | 0100 1111 | 4FH |
| Add the contents of register B to the contents of the accumulator. | ADD | B | 1000 0000 | 80H |
| Invert (complement) each bit in the accumulator. | CMA | | 0010 1111 | 2FH |

## TWO-BYTE INSTRUCTIONS

In a 2-byte instruction, the first byte specifies the operation code and the second byte specifies the operand. For example:

| Task | Opcode | Operand | Binary Code | Hex Code | |
|---|---|---|---|---|---|
| Load an 8-bit data byte in the accumulator. | MVI | A,32H | 0011 1110 | 3E | First Byte |
| | | | 0011 0010 | 32 | Second Byte |
| Load an 8-bit data byte in register B. | MVI | B,F2H | 0000 0110 | 06 | First Byte |
| | | | 1111 0010 | F2 | Second Byte |

These instructions would require two memory locations each to store the binary codes. The data bytes 32H and F2H are selected arbitrarily as examples.

## THREE-BYTE INSTRUCTIONS

In a 3-byte instruction, the first byte specifies the opcode, and the following two bytes specify the 16-bit address. Note that the second byte is the low-order address and the third byte is the high-order address. For example:

| Task | Opcode | Operand | Binary Code | Hex Code* | |
|---|---|---|---|---|---|
| Load contents of memory 2050H into A. | LDA | 2050H | 0011 1010 | 3A | First Byte |
| | | | 0101 0000 | 50 | Second Byte |
| | | | 0010 0000 | 20 | Third Byte |
| Transfer the program sequence to memory location 2085H. | JMP | 2085H | 1100 0011 | C3 | First Byte |
| | | | 1000 0101 | 85 | Second Byte |
| | | | 0010 0000 | 20 | Third Byte |

These instructions would require three memory locations each to store the binary codes.

| Mnemonics | Hex Code | |
|---|---|---|
| MVI A,32H | 3E | 2-byte instruction |
| | 32 | |
| MVI B,48H | 06 | 2-byte instruction |
| | 48 | |
| ADD B | 80 | 1-byte instruction |
| OUT 01H | D3 | 2-byte instruction |
| | 01 | |
| HLT | 76 | 1-byte instruction |

| Mnemonics | Hex Code | Memory Contents | Memory Address |
|---|---|---|---|
| MVI A,32H | 3E | 0 0 1 1 1 1 1 0 | 2000 |
| | 32 | 0 0 1 1 0 0 1 0 | 2001 |
| MVI B,48H | 06 | 0 0 0 0 0 1 1 0 | 2002 |
| | 48 | 0 1 0 0 1 0 0 0 | 2003 |
| ADD B | 80 | 1 0 0 0 0 0 0 0 | 2004 |
| OUT 01H | D3 | 1 1 0 1 0 0 1 1 | 2005 |
| | 01 | 0 0 0 0 0 0 0 1 | 2006 |
| HLT | 76 | 0 1 1 1 1 1 1 0 | 2007 |


# 8085 MICROPROCESSOR INSTRUCTION SET HAS 74 DIFFERENT INSTRUCTION


# 74 INSTRUCTIONS GIVES A TOTAL OF 246 BIT PATTERNS (EACH PATTERN IS OF 8 BIT)

# Instruction Set of 8085

| Notation | Meaning |
|---|---|
| M | Memory location pointed by HL register pair |
| r | 8-bit register |
| rp | 16-bit register pair |
| rs | Source register |
| rd | Destination register |
| addr | 16-bit address / 8-bit address |

# Data Transfer Group

1. **MVI r, data (8)** This instruction directly loads a specified register with an 8-bit data given within the instruction. The register r is an 8-bit general purpose register such as A, B, C, D, E, H and L.

**Operation** : r ← 8-bit data (byte)

**Example** :

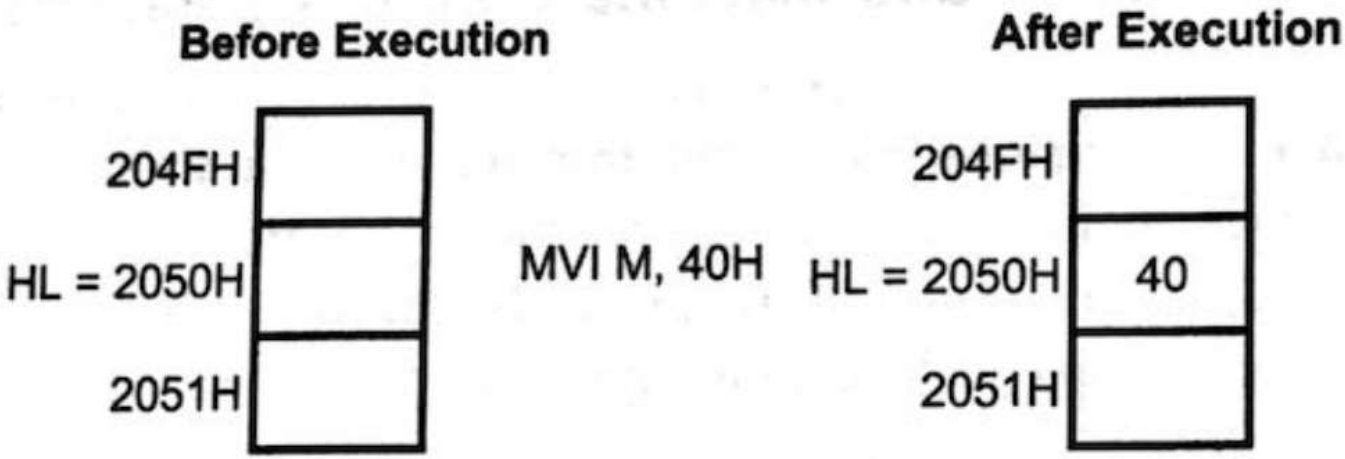
MVI B, 60H ; This instruction will load 60H directly into the B register.

2. **MVI M, data (8)** This instruction directly loads an 8-bit data given within the instruction into a memory location. The memory location is specified by the contents of HL register pair.

**Operation** : M ← byte or (HL) ← byte

**Example** : H = 20H and L = 50H

MVI M, 40H ; This instruction will load 40H into memory whose address is 2050H.



**3. MOV rd, rs** This instruction copies data from the source register into destination register. The rs and rd are general purpose registers such as A, B, C, D, E, H and L. The contents of the source register remain unchanged after execution of the instruction.

**Operation** : rd ← rs

**Example** : A = 20H

MOV B, A ; This instruction will copy the contents of register A (20H) into register B.

**4. MOV M, rs** This instruction copies data from the source register into memory location pointed by the HL register pair. The rs is an 8-bit general purpose register such as A, B, C, D, E, H and L.

**Operation** : (HL) ← rs

**Example** : If HL = 2050H, B = 30H.

MOV M, B ; This instruction will copy the contents of B register (30H) into the memory location whose address is specified by HL (2050H).

**5. MOV rd, M** This instruction copies data from memory location whose address is specified by HL register pair into destination register. The contents of the memory location remain unchanged. The rd is an 8-bit general purpose register such as A, B, C, D, E, H and L.

**Operation** : rd ← (HL)

**Example** : HL = 2050H, contents at 2050H memory location = 40H

MOV C, M ; This instruction will copy the contents of memory location pointed by HL register pair (40H) into the C register.

6. **LXI rp, data (16)** This instruction loads immediate 16 bit data specified within the instruction into register pair or stack pointer. The rp is 16-bit register pair such as BC, DE, HL or 16-bit stack pointer.

**Operation** : rp ← data (16)

**Example** :

LXI B,1020H ; This instruction will load 10H into B register and 20H into C register.

**7. STA addr** This instruction stores the contents of A register into the memory location whose address is directly specified within the instruction. The contents of A register remain unchanged.

**Operation** : (addr) ← A

**Example** : A = 50H

STA 2000H ; This instruction will store the contents of A register (50H) to memory location 2000H.

**8. LDA addr** This instruction copies the contents of the memory location whose address is given within the instruction into the accumulator. The contents of the memory location remain unchanged.

**Operation :** $A \leftarrow (addr)$

**Example :** (2000H) = 30H

LDA 2000H ; This instruction will copy the contents of memory location 2000H i.e. data 30H into the A register

**9. SHLD addr** This instruction stores the contents of L register in the memory location given within the instruction and contents of H register at address next to it. This instruction is used to store the contents of H and L registers directly into the memory. The contents of the H and L registers remain unchanged.

**Operation** : (addr) ← L and (addr + 1) ← H

**Example** : H = 30H, L = 60H

SHLD 2500H ; This instruction will copy the contents of L register at address 2500H and the contents of H register at address 2501H.

**10. LHLD addr** This instruction copies the contents of the memory location given within the instruction into the L register and the contents of the next memory location into the H register.

**Operation** : L ← (addr), H ← (addr + 1)

**Example** : (2500H) = 30H, (2501H) = 60H

LHLD 2500H ; This instruction will copy the contents of memory location 2500H i.e. data 30H into the L register and the contents at memory location 2501H i.e. data 60H into the H register.

**11. STAX rp** This instruction copies the contents of accumulator into the memory location whose address is specified by the specified register pair. The rp is BC or DE register pair. This register pair is used as a memory pointer. The contents of the accumulator remain unchanged.

**Operation** : (rp) ← A

**Example** : BC = 1020H, A = 50H

STAX B ; This instruction will copy the contents of A register (50H) to the memory location specified by BC register pair (1020H).

**12. LDAX rp** This instruction copies the contents of memory location whose address is specified by the register pair into the accumulator. The rp is BC or DE register pair. The register pair is used as a memory pointer.

**Operation** : $A \leftarrow (rp)$

**Example** : DE = 2030H, (2030H) = 80H

LDAX D This instruction will copy the contents of memory location specified by DE register pair (80H) into the accumulator.

**13. XCHG** This instruction exchanges the contents of the register H with that of D and of L with that of E.

**Operation** : $H \leftrightarrow D$ and $L \leftrightarrow E$

**Example** : DE = 2040H, HL = 7080H

```
XCHG    ; This instruction will load the data into registers as follows
          H = 20H, L = 40H, D = 70H and E = 80
```

# Arithmetic Group

1. **ADD r** — This instruction adds the contents of the specified register to the contents of accumulator and stores result in the accumulator. The r is 8-bit general purpose register such as A, B, C, D, E, H and L.

**Operation** : $A \leftarrow A + r$

**Example** : A = 20H, C = 30H.

ADD C ; This instruction will add the contents of C register, i.e. data 30H to the contents of accumulator, i.e. data 20H and it will store the result 50H in the accumulator.

**2. ADD M** This instruction adds the contents of the memory location pointed by HL register pair to the contents of accumulator and stores result in the accumulator. The HL register pair is used as a memory pointer. This instruction affects all flags.

**Operation** : $A \leftarrow A + M$

**Example** : A = 20H, HL = 2050H, (2050H) = 10H

ADD M ; This instruction will add the contents of memory location pointed by HL register pair, 2050H i.e. data 10H to the contents of accumulator i.e. data 20H and it will store the result, 30H in the accumulator.

**3. ADI data (8)** This instruction adds the 8 bit data given within the instruction to the contents of accumulator and stores the result in the accumulator.

**Operation** : $A \leftarrow A + \text{data (8)}$

**Example** : $A = 50H$

ADI 70H ; This instruction will add 70H to the contents of the accumulator (50H) and it will store the result in the accumulator (C0H).

**4. ADC r**

This instruction adds the contents of specified register to the contents of accumulator with carry. This means, if the carry flag is set by some previous operation, it adds 1 and the contents of the specified register to the contents of accumulator, else it adds the contents of the specified register only. The r is 8-bit general purpose register such as A, B, C, D, E, H and L.

**Operation** : $A \leftarrow A + r + CY$

**Example** : Carry flag = 1, A = 50H, C = 20H

ADC C ; This instruction will add the contents of C (20H) register to the contents of accumulator (50H) with carry (1) and it will store result, 71H (50H + 20H + 1 = 71H) in the accumulator

**5. ADC M** This instruction adds the contents of memory location pointed by HL register pair to the contents of accumulator with carry and stores the result in the accumulator. HL register pair is used as a memory pointer.

**Operation** : $A \leftarrow A + M + CY$

**Example** : Carry flag = 1, HL = 2050H, A = 20H, (2050H) = 30H.

ADC M ; This instruction will add the contents of memory location pointed by HL register pair, 2050H, i.e. data 30H to the contents of accumulator, i.e. data 20H with carry flag (1). It will store the result (30+20+1=51H) in the accumulator.

**6. ACI data (8)** This instruction adds 8 bit data given within the instruction to the contents of accumulator with carry and stores result in the accumulator.

**Operation** : $A \leftarrow A + \text{data (8)} + CY$

**Example** : A = 30H, Carry flag = 1

ACI 20H ; This instruction will add 20H to the contents of accumulator, i.e. data 30H with carry (1) and stores the result, 51H (30 + 20 + 1 = 51H) in the accumulator.

**7. DAD rp** This instruction adds the contents of the specified register pair to the contents of the HL register pair and stores the result in the HL register pair. The rp is 16-bit register pair such as BC, DE, HL or stack pointer. Only higher order register is to be specified for register pair within the instruction.

**Operation** : HL ← HL + rp

**Example** : DE = 1020H, HL = 2050H

DAD D ; This instruction will add the contents of DE register pair, 1020H to the contents of HL register pair, 2050H. It will store the result, 3070H in the HL register pair.

**8. SUB r** This instruction subtracts the contents of the specified register from the contents of the accumulator and stores the result in the accumulator. The register r is 8-bit general purpose register such as A, B, C, D, E, H and L.

**Operation** : $A \leftarrow A - r$

**Example** : A = 50H, B = 30H.

```
SUB B    ; This instruction will subtract the contents of B register (30H) from
           the contents of accumulator (50H) and stores the result (20H) in
           the accumulator.
```

**9. SUB M** This instruction subtracts the contents of the memory location pointed by HL register pair from the contents of accumulator and stores the result in the accumulator. The HL register pair is used as a memory pointer.

**Operation** : $A \leftarrow A - M$

**Example** : HL = 1020H, A = 50H, (1020H) = 10H

```
SUB M    ; This instruction will subtract the contents of memory location
           pointed by HL register pair, 1020H, i.e. data 10H from the contents
           accumulator, i.e. data 50H and stores the result (40H) in
           accumulator.
```

**10. SUI data (8)** This instruction subtracts an 8 bit data given within the instruction from the contents of the accumulator and stores the result in the accumulator.

**Operation** : $A \leftarrow A - \text{data (8)}$

**Example** : A = 40H,

SUI 20H ; This instruction will subtract 20H from the contents of accumulator (40H). It will store the result (20H) in the accumulator.

**11. SBB r** This instruction subtracts the specified register contents and borrow flag from the accumulator contents. This means, if the carry flag (borrow for subtraction) is set by some previous operation, it subtracts 1 and the contents of the specified register from the contents of accumulator, else it subtracts the contents of the specified register only. The register r is 8-bit register such as A, B, C, D, E, H and L.

**Operation** : $A \leftarrow A - r - CY$

**Example** : Carry flag = 1, C = 20H, A = 40H

SBB C ; This instruction will subtract the contents of C register (20H) and carry flag (1) from the contents of accumulator (40H).It will store the result (40H – 20H – 1 = 1FH) in the accumulator.

**12. SBB M** This instruction subtracts the contents of memory location pointed by HL register pair from the contents of accumulator and borrow flag and stores the result in the accumulator.

**Operation** : $A \leftarrow A - M - CY$

**Example** : Carry flag = 1, HL = 2050H, A = 50H, (2050H) = 10H.

```
SBB M    ; This instruction will subtract the contents of memory location;
           pointed by HL register pair, 2050H, i.e. data 10H and borrow
           (Carry flag=1) from the contents of accumulator (50H) and
           stores the result 3FH in the accumulator (50 – 10 – 1 = 3F).
```

**13. SBI data (8)** This instruction subtracts 8 bit data given within the instruction and borrow flag from the contents of accumulator and stores the result in the accumulator.

**Operation** : $A \leftarrow A - data(8) - CY$

**Example** : Carry flag = 1, A = 50H

```
SBI 20H     ; This instruction will subtract 20H and the carry flag (1) from the
              contents of the accumulator (50H). It will store the result
              (50H – 20H – 1 = 2FH) in the accumulator.
```

**14. DAA**

This instruction adjusts accumulator to packed BCD (Binary Coded Decimal) after adding two BCD numbers.

Instruction works as follows :

1. If the value of the low - order four bits ($D_3$-$D_0$) in the accumulator is greater than 9 or if auxiliary carry flag is set, the instruction adds 6 (06) to the low-order four bits.

2. If the value of the high-order four bits ($D_7$-$D_4$) in the accumulator is greater than 9 or if carry flag is set, the instruction adds 6 (60) to the high-order four bits.

**Example :**

If, A = 0011 1001 = 39 BCD

and C = 0001 0010 = 12 BCD then

```
ADD C      ; Gives A = 0100 1011 = 4BH
DAA        ; adds 0110 because 1011>9,
           ; A=0101 0001 = 51 BCD
```

If A = 1001 0110 = 96 BCD

and D = 0000 0111 = 07 BCD then

```
ADD D      ; Gives A = 1001 1101 = 9DH
DAA        ; adds 0110 because 1101 > 9,
                A = 1010 0011 = A3H,
           1010 > 9 so adds 0110 0000,
      A = 0000 0011 = 03 BCD, CF = 1.
```

**15. INR r** This instruction increments the contents of specified register by 1. The result is stored in the same register. The register r is 8-bit general purpose register such as A, B, C, D, E, H and L.

**Operation** : $r \leftarrow r + 1$

**Example** : B = 10H

INR B ; This instruction will increment the contents of B register (10H) by one and stores the result (10+1 = 11H) in the same i.e. B register.

**16. INR M** This instruction increments the contents of memory location pointed by HL register pair by 1. The result is stored at the same memory location. The HL register pair is used as a memory pointer.

**Operation** : $M \leftarrow M + 1$

**Example** : HL = 2050H, (2050H) = 30H

INR M ; This instruction will increment the contents of memory location pointed by HL register pair, 2050H, i.e. data 30H by one. It will store the result (30 + 1 = 31H) at the same place.

**17. INX rp** This instruction increments the contents of register pair by one. The result is stored in the same register pair. The rp is register pair such as BC, DE, HL or Stack Pointer (SP).

**Operation** : rp ← rp + 1

**Example** : HL = 10FFH

INX H ; This instruction will increment the contents of HL register pair (10FFH) by one. It will store the result (10FF + 1 = 1100H) in the same i.e. HL register pair.

**18. DCR r** This instruction decrements the contents of the specified register by one. It stores the result in the same register. The register r is 8-bit general purpose register such as A, B, C, D, E, H and L.

**Operation** : $r \leftarrow r - 1$

**Example** : E = 20H

DCR E ; This instruction will decrement the contents of E register (20H) by one. It will store the result (20 – 1 = 1FH) in the same, i.e. E register.

**19. DCR M** This instruction decrements the contents of memory location pointed by HL register pair by 1. The HL register pair is used as a memory pointer. The result is stored in the same memory location.

**Operation** : $M \leftarrow M - 1$

**Example** : HL = 2050H, (2050H) = 21H

DCR M ; This instruction will decrement the contents of memory location pointed by HL register pair, 2050H, i.e. data 21H by one. It will store the result (21 – 1 = 20H) in the same memory location.

**20. DCX rp** This instruction decrements the contents of register pair by one. The result is stored in the same register pair. The rp is register pair such as BC, DE, HL or Stack Pointer (SP). Only higher order register is to be specified within the instruction.

**Operation** : rp ← rp – 1

**Example** : DE = 1020H

DCX D ; This instruction will decrement the contents of DE register pair (1020H) by one and store the result (1020 – 1 = 101FH) in the same, DE register pair.

# Logic Group

**1. ANA r**

This instruction logically ANDs the contents of the specified register with the contents of accumulator and stores the result in the accumulator. Each bit in the accumulator is logically ANDed with the corresponding bit in register r, i.e. $D_0$ bit in A with $D_0$ bit in register r, $D_1$ in A with $D_1$ in r and so on upto $D_7$ bit. The register r is 8-bit general purpose register such as A, B, C, D, E, H and L.

**Operation** : $A \leftarrow A \wedge r$

**Example** : A = 10101010 (AAH), B = 00001111 (0FH)

ANA B ; This instruction will logically AND the contents of B register with the contents of accumulator. It will store the result (0AH) in the accumulator.

```
1010  1010
0000  1111
---------
0000  1010  = 0AH
```

**2. ANA M** This instruction logically ANDs the contents of memory location pointed by HL register pair with the contents of accumulator. The result is stored in the accumulator. The HL register pair is used as a memory pointer.

**Operation** : $A \leftarrow A \wedge M$

**Example** : A = 01010101 = (55H), HL = 2050H(2050H) → 10110011 = (B3H)

ANA M ; This instruction will logically AND the contents of memory location pointed by HL register pair (B3H) with the contents of accumulator (55H). It will store the result (11H) in the accumulator.

```
0101  0101
1011  0011
---------
0001  0001  = 11H
```

**3. ANI data** This instruction logically ANDs the 8 bit data given in the instruction with the contents of the accumulator and stores the result in the accumulator.

**Operation** : $A \leftarrow A \wedge$ data (8)

**Example** : A = 1011 0011 = (B3H)

ANI 3FH ; This instruction will logically AND the contents of accumulator (B3H) with 3FH. It will store the result (33H) in the accumulator.

```
1 0 1 1   0 0 1 1
0 0 1 1   1 1 1 1
-------------------
0 0 1 1   0 0 1 1  = 33H
```

**4. XRA r**

This instruction logically XORs the contents of the specified register with the contents of accumulator and stores the result in the accumulator. The register r is 8-bit general purpose register such as A, B, C, D, E, H and L.

**Operation** : $A \leftarrow A \oplus r$

**Example** : A = 1010 1010 (AAH), C = 0010 1101 (2DH)

XRA C ; This instruction will logically XOR the contents of C register with the contents of accumulator. It will store the result (87H) in the accumulator.

```
1 0 1 0   1 0 1 0
0 0 1 0   1 1 0 1
-----------------
1 0 0 0   0 1 1 1 = (87H)
```

**5. XRA M** This instruction logically XORs the contents of memory location pointed by HL register pair with the contents of accumulator. The HL register pair is used as a memory pointer.

**Operation** : $A \leftarrow A \oplus M$

**Example** : **A = 0101 0101 = (55H), HL = 2050H (2050H) → 1011 0011 =(B3H)**

**XRA M** ; **This instruction will logically XOR the contents of memory location pointed by HL register pair (2050H) i.e. data B3H with the contents of accumulator (55H). It will store the result (E6H) in the accumulator.**

```
0101  0101
1011  0011
---------
1110  0110 = E6H
```

**6. XRI data** This instruction logically XORs the 8 bit data given in the instruction with the contents of the accumulator and stores the result in the accumulator.

**Operation** : A ← A ⊕ data

**Example** : A = 10110011 = (B3H)

XRI 39H ; This instruction will logically XOR the contents of accumulator (B3H) with 39H. It will store the result (8AH) in the accumulator.

```
1011  0011
0011  1001
---------
1000  1010 = 8AH
```

**7. ORA r** This instruction logically ORs the contents of specified register with the contents of accumulator and stores the result in the accumulator. Each bit in the accumulator is ORed with corresponding bit in register r. i.e. $D_0$ bit in accumulator is ORed with $D_0$ bit in register r, $D_1$ in A with $D_1$ in r and so on upto $D_7$ bit. The register r is 8-bit general purpose register such as A, B, C, D, E, H and L.

**Operation** : $A \leftarrow A \vee r$

**Example** : A = 1010 1010 (AAH), B = 0001 0010 (12H)

ORA B ; This instruction will logically OR the contents of B register with the contents of accumulator. It will store the result (BAH) in the accumulator.

```
1010  1010
0001  0010
---------
1011  1010 = BAH
```

**8. ORA M** This instruction logically ORs the contents of memory location pointed by HL register pair with the contents of accumulator. The result is stored in the accumulator. The HL register pair is used as a memory pointer.

**Operation** : $A \leftarrow A \vee M$

**Example** : A = 0101 0101 = (55H) HL = 2050H

(2050H) $\rightarrow$ 1011 0011 = (B3H)

```
ORA M          ;
0 1 0 1   0 1 0 1
1 0 1 1   0 0 1 1
-----------------
1 1 1 1   0 1 1 1 = F7H
```

This instruction will logically OR the contents of memory location pointed by HL register pair (B3H) with the contents of accumulator (55H). It will store the result (F7H) in the accumulator.

**9. ORI data** This instruction logically ORs the 8 bit data given in the instruction with the contents of the accumulator and stores the result in the accumulator.

**Operation** : A $\vee$ data (8)

**Example** : A = 1011 0011 = (B3H)

```
ORI 08H      ;  This instruction will logically OR the contents of accumulator (B3H)
1 0 1 1  0 0 1 1   with 08H. It will store the result (BBH) in the accumulator.
0 0 0 0  1 0 0 0
-----------------
1 0 1 1  1 0 1 1 = BBH
```

**10. CMP r** This instruction subtracts the contents of the specified register from contents of the accumulator and sets the condition flags as a result of the subtraction. It sets zero flag if A = r and sets carry flag if A < r. The register r is 8-bit general purpose register such as A, B, C, D, E, H and L.

**Operation** : A – r

**Example** : A = 1011 1000 (B8H) and D = 1011 1001 (B9H)

CMP D ; This instruction will compare the contents of D register with the contents of accumulator. Here A < D so carry flag will set after the execution of the instruction.

**11. CMP M** This instruction subtracts the contents of the memory location specified by HL register pair from the contents of the accumulator and sets the condition flags as a result of subtraction. It sets zero flag if A = M and sets carry flag if A < M. The HL register pair is used as a memory pointer.

**Operation** : A – M

**Example** : A = 1011 1000 (B8H), HL = 2050H and (2050H) = 1011 1000 (B8H)

```
CMP M    ; This instruction will compare the contents of memory location
           (B8H) and the contents of accumulator. Here A = M so zero flag
           will set after the execution of the instruction.
```

**12. CPI data** This instruction subtracts the 8 bit data given in the instruction from the contents of the accumulator and sets the condition flags as a result of subtraction. It sets zero flag if A = data and sets carry flag if A < data.

**Operation** : A – data (8)

**Example** : A = 1011 1010 = (BAH)

CPI 30H ; This instruction will compare 30H with the contents of accumulator (BAH). Here A > data so zero and carry both flags will reset after the execution of the instruction.

**13. STC** This instruction sets carry flag = 1

**Operation** : $CY \leftarrow 1$

**Example** : Carry flag = 0

```
STC     ;  This instruction will set the carry flag = 1
```

**14. CMC** This instruction complements the carry flag.

**Operation** : $CY \leftarrow \overline{CY}$

**Example** : Carry flag = 1

CMC ; This instruction will complement the carry flag i.e. carry flag = 0

**15. CMA** This instruction complements each bit of the accumulator.

**Operation** : $A \leftarrow \overline{A}$

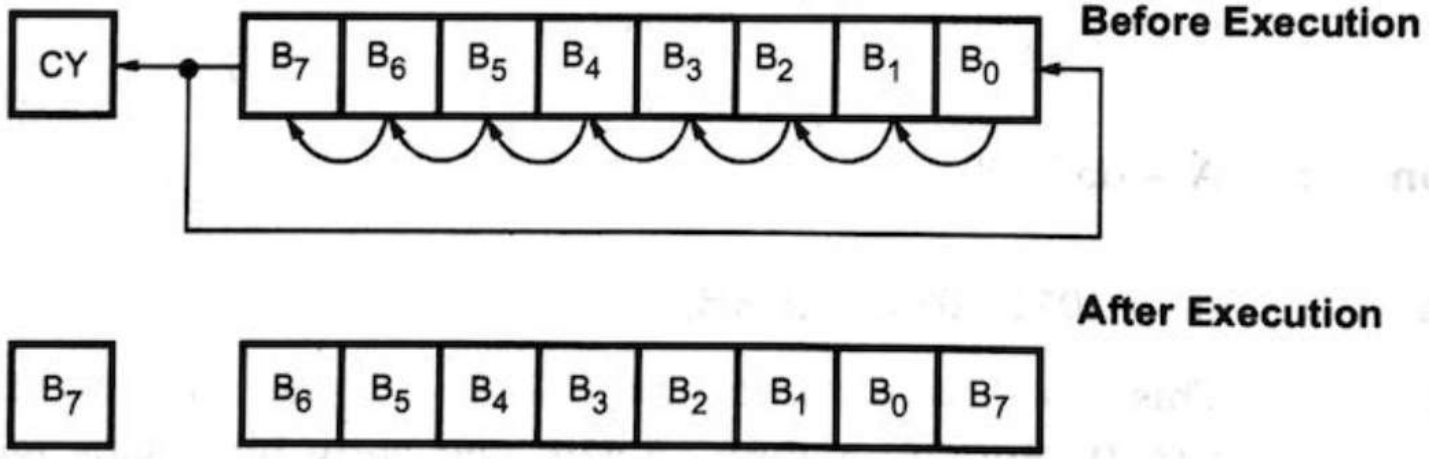
**Example** : A = 1000 1000 = 88H

CMA ; This instruction will complement each bit of accumulator A = 0111 0111 = 77H

# Rotate Group

**1. RLC** This instruction rotates the contents of the accumulator left by one position. Bit $B_7$ is placed in $B_0$ as well as in CY.

**Operation :**

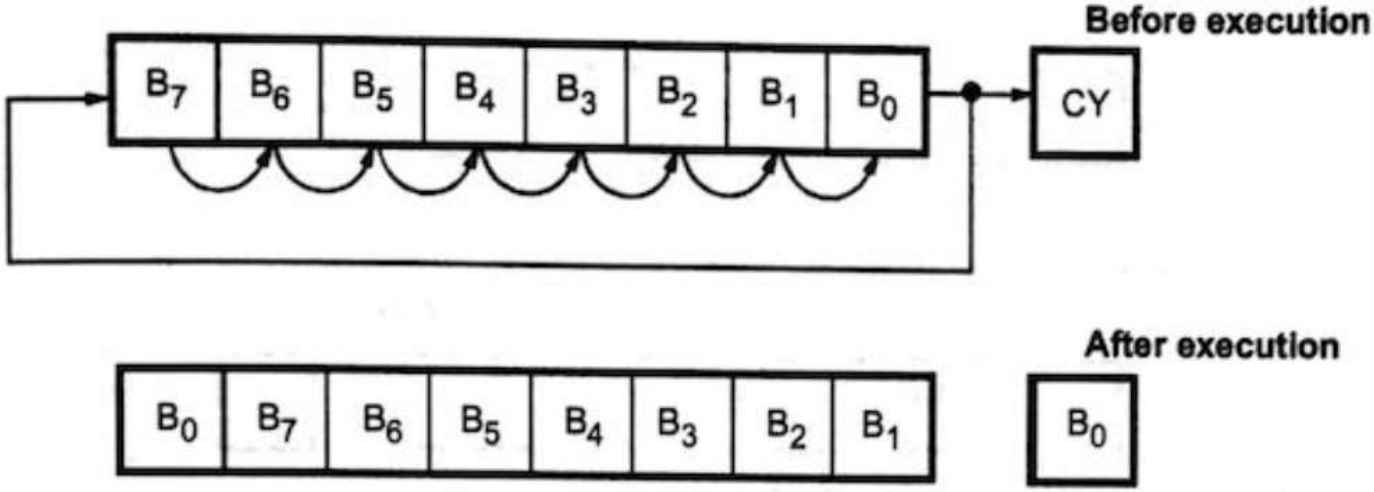



**Example** : A = 01010111 (57H) and CY = 1

RLC ; After execution of the instruction the accumulator contents will be (1010 1110) AEH and carry flag will reset.

**2. RRC** This instruction rotates the contents of the accumulator right by one position. Bit $B_0$ is placed in $B_7$ as well as in CY.

**Operation :**

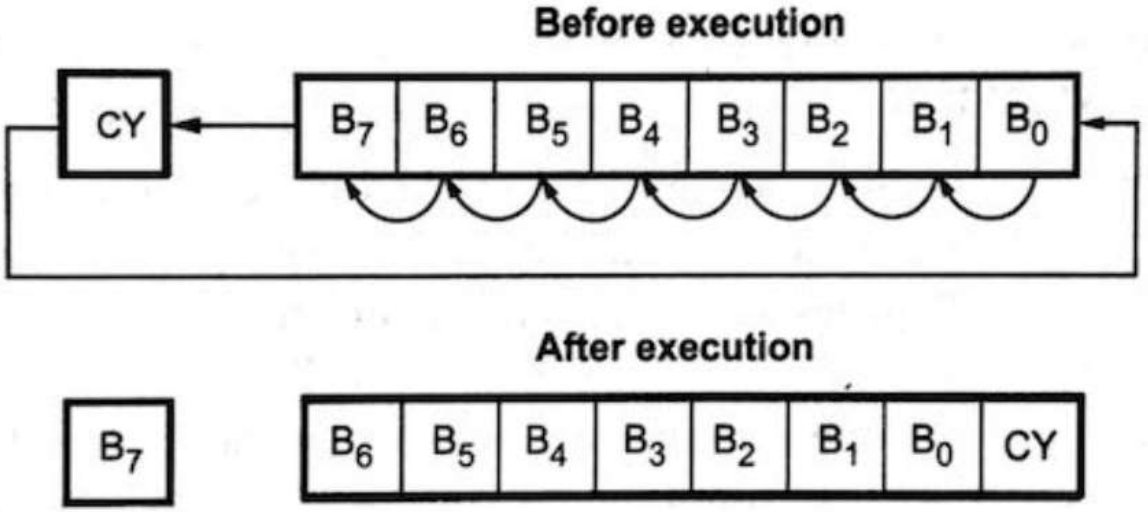



**Example** : A = 1001 1010 (9AH) and CY = 1

RRC ; After execution of the instruction the accumulator contents will be (0100 1101) 4DH and carry flag will reset.

**3. RAL** This instruction rotates the contents of the accumulator left by one position. Bit $B_7$ is placed in CY and CY is placed in $B_0$.

**Operation :**

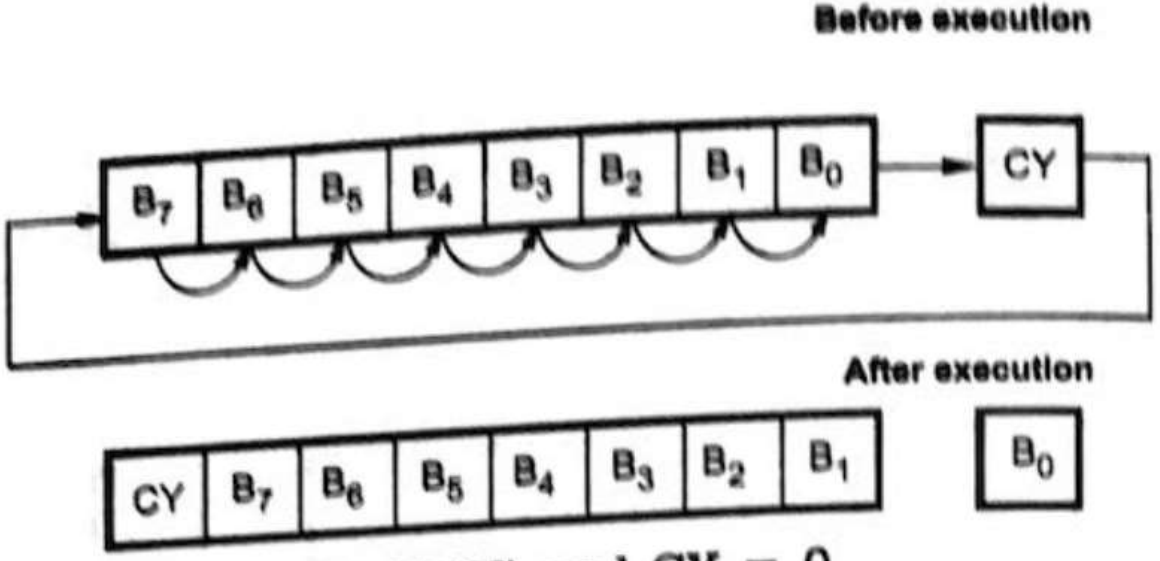



**Example** : A = 10101101 (ADH) and CY = 0

RAL ; After execution of the instruction accumulator contents will be (0101 1010) 5AH and carry flag will set.

**4. RAR** This instruction rotates the contents of the accumulator right by one position. Bit $B_0$ is placed in CY and CY is placed in $B_7$.

**Operation :**




**Example** : A = 1010 0011 (A3H) and CY = 0

RAR ; After execution of the instruction accumulator contents will be (0101 0001) 51H and carry flag will set.