2. INSTRUCTION SET OF 8085

I Data Transfer Group:

1) MOV $r_d$, $r_s$ : [MOVE REGISTER]
   
   $[r_d] \leftarrow [r_s]$  
   
   Addressing : Register addressing  
   Group : Data transfer group  
   Bytes : 1 byte  
   
   Comment: This instruction will copy destination register with the content of source register. The content of source register are not altered i.e. they remain unchanged.  
   $r_d$ and $r_s$ can be one of registers A, B, C, D, E, H, L.  
   Example : Let $[A] = 05H$ and $[B] = 55H$  
   Instruction: MOV A, B  
   After execution: $[A] = 55H$ and $[B] = 55H$  

2) MOV $r$, M : [MOVE FROM MEMORY]
   
   $[r] \leftarrow [[H-L]]$  
   
   Addressing : Register indirect addressing  
   Group : Data transfer group  
   Bytes : 1 byte  
   
   Comment: This instruction will load destination register with the content of memory location, whose address is stored in H-L register pair. The content of memory location are not altered. $r$ can be any one of register A, B, C, D, E.  
   Instruction: MOV B, M  
   After execution: $[B] = 35H$  

   $[CFFF] = 35H$  

3) MOV M, $r$ : [MOVE TO MEMORY]
   
   $[[H-L]] \leftarrow [r]$  
   
   Addressing : Register indirect  
   Group : Data transfer group  
   Bytes : 1 byte  
   
   Comment: This instruction will copies the content of register $r$ to the memory location, whose address is placed in H-L register pair. $r$ can be any one of the A, B, C, D, E.  
   Example : Let $[HL] = F000H$ and  
   $[F000] = 40H$ and $[C] = FAH$ then  
   Instruction: MOV M, C  
   After execution: $[C] = FAH$
4) **MVI r, data:** [MOVE IMMEDIATE 8-BIT]
   : [r] ← data (second byte)
   **Addressing:** Immediate addressing
   **Group:** Data transfer group
   **Bytes:** 2 byte
   **Comments:** This instruction will load the register r with 8-bit immediate data specified in second byte of instruction.
   **Example:** Instruction: MVI A, 35H
   This instruction will load accumulator with immediate data 35H.

5) **MVI M, data:** [MOVE IMMEDIATE 8-BIT]
   : [H-L] ← data (second byte)
   **Addressing:** Immediate / Register indirect address
   **Group:** Data transfer group
   **Bytes:** 2 byte
   **Comment:** This instruction will load the memory location, whose address is stored in H-L pair with 8-bit immediate data specified in the second byte of instruction.
   **Example:** Let [H] [L] = D000H
   Instruction: MVI M, 35H
   Above instruction will load memory location D000H with immediate data 35H.

6) **LXI r_p, 16-bit data:** [LOAD REGISTER PAIR IMMEDIATE]
   : [r_h] ← 16 bit data i.e. [r_h] ← byte 3, [r_i] ← byte 2
   **Addressing:** Immediate
   **Group:** Data transfer group
   **Bytes:** 3 byte
   **Comment:** The byte 3 of instruction is moved into high order register (r_h) of register pair rp and byte 2 is moved into low order register (r_i) of register pair. The register pairs can be BC, DE, HL or SP. [SP (stack pointer) is not a valid register pair, but it can be used in LXI instruction].
   **Example:** LXI H, 3500 H
   This instruction will load H-L pair with 3500 H. 35 H will be loaded in high order register (H) and 00H will be loaded in low order register (L).

7) **LDA addr:** [LOAD ACCUMULATOR DIRECT]
   : [A] ← [byte 3] [ byte 2]
   **Addressing:** Direct addressing mode
   **Group:** Data transfer group
   **Bytes:** 3 byte
   **Comment:** This instruction will load accumulator with content of memory location, whose address is given in the instruction itself. The content of memory location are note altered.
Example: Let [C500] = 26 H

Instruction: LDA C500
After execution: [A] = 26 H
[C500] = 26 H

8) **STA addr**: [STORE ACCUMULATOR DIRECT]

   : [[byte 3] [byte 2]] ← [A]

Addressing: Direct addressing
Group: Data transfer group
Bytes: 3 byte

**Comment**: This instruction will load content of accumulator into the memory location, whose address is specified in the instruction. The content of accumulator are not altered.

Example: Let [A] = 35 H

Instruction: STA C500 H
After execution: [C500] = 35 H
[A] = 35 H

9) **LHLD addr**: [LOAD H AND L REGISTER DIRECT]

   : [L] ← [[byte 3] [byte 2]]
   [H] ← [[byte 3] [byte 2] + 1]

Addressing: Direct addressing
Group: Data transfer group
Bytes: 3 byte

**Comment**: In this instruction, the first byte gives the opcode and second and third byte give 16-bit address of memory location in usual convention. The content of memory location whose address is specified in the instruction are loaded into register L and the content of next memory location loaded in register H.

For example:

Let memory location 2100 H contains 31 H and 2101 H contains 52 H then after execution of instruction –

LHLD 2100 H

Register H will contain 52 H and register L will contain 31 H

10) **SHLD addr**: [STORE H AND L REGISTERS DIRECT]

   : [[byte 3] [byte 2]] ← [L]
   [[byte 3] [byte 2] + 1] ← [H]
Addressing: Direct addressing
Group: Data transfer group
Bytes: 3 byte

Comment: The content of register L are transferred to the memory location whose address is specified by byte 2 and byte 3 of the instruction. The contents of register H are moved to succeeding memory location.

Example: Let [H] = 32 H and [L] = 35 H
          Instruction: SHLD 2100 H
          After execution: [2100] = 35 H
                          [2101] = 32 H

11) LDAX rp: [LOAD ACCUMULATOR INDIRECT]
    : [A] ← [rp]
Addressing: Register indirect
Group: Data transfer group
Bytes: 1 byte

Comment: The content of memory location, whose address is stored in register pair rp are loaded into accumulator. The content of memory location remains unchanged. rp can be B (i.e. B and C) or D. (i.e. D and E)

          Instruction: LDAX B
          After execution: [A] = 33 H

12) STAX rp: [STORE ACCUMULATOR INDIRECT]
    : [rp] ← [A]
Addressing: Register indirect addressing
Group: Data transfer group
Bytes: 1 byte

Comment: The content of accumulator are transferred to the memory location whose address is stored in register pair rp. The valid register pairs are B (i.e. B and C) and D (i.e. D and E)

          Instruction: STAX D
          After execution: [2525] = 55 H

13) XCHG: [EXCHANGE H AND L WITH D AND E]
    : [H] ↔ [D]
    [L] ↔ [E]
Addressing: Register
Group: Data transfer group
Bytes: 1 byte

Comment: The content of register H are exchanged with that of register D and the content of register L are exchanged with that of register E.
Example: Let \([H] = 23\ H, [L] = 32\ H, [D] = 53\ H\) and \([E] = 55\ H\)

Instruction: XCHG

After execution: \([H] = 53\ H\) and \([L] = 55\ H\)

\([D] = 23\ H\) and \([E] = 32\ H\)

II Arithmetic Group:

1) ADD \(r\) : [ADD REGISTER]

\([A] \leftarrow [A] + [r]\)

Addressing: Register addressing

Group: Arithmetic group

Bytes: 1 byte

Comment: The content of register \(r\) are added to the content of accumulator. The result is stored in accumulator. All the flags may be affected.

Example: Let \([D] = 35\ H\) and \([A] = 05H\)

Instruction: ADD D

Addition: \(35\ H = 00110101\)

\(+05\ H = 00000101\)

\(3AH = 00111010\)

\(S = 0, Z = 0,\ Ac = 0\)

\(P = 1, Cy = 0\)

After execution:

\([A] = 3AH\)

Flag Register =

\[
\begin{array}{cccccc}
0 & 0 & - & 0 & - & 1 & - & 0
\end{array}
\]

\([D] = 35\ H\)

2) ADD \(M\) : [ADD MEMORY CONTENT TO ACCUMULATOR]

\([A] \leftarrow [A] + [H][L]\)

Addressing: Register indirect addressing

Group: Arithmetic group

Bytes: 1 byte

Comment: The content of the content of memory location, whose address is stored in H-L pair are added with accumulator. The result is placed in accumulator. All flags may be affected.

Example: Let \([H-L] = D000\ H, [D000] = 51\ H\) and \([A] = 35\ H\)

Instruction: ADD M

After execution: \([A] = 86\ H\) and \([D000] = 51\ H\)

3) ADD data : [ADD IMMEDIATE TO ACCUMULATOR]

\([A] \leftarrow [A] + \text{data (byte 2)}\)

Addressing: Immediate addressing

Group: Arithmetic group
Bytes : 2 byte

**Comment:** This instruction adds the 8-bit immediate data specified in second byte of instruction to the content of accumulator. All flags may be affected.

**Example:** Let, [A] = EAH

**Instruction:** ADI 15 H

**Addition:**
- (A): EAH = 1 1 1 1 0 1 0 1 0
- Data: +15 H = 0 0 0 1 0 1 0 1
- FFH = 1 1 1 1 1 1 1 1

**Flags**
- S = 1, Z = 0, Ac = 0
- P = 1, Cy = 0

**After execution:** [A] = FFH

4) **ADC r : [ADD REGISTER TO ACCUMULATOR WITH CARRY]**

: [A] ← [A] + [r] + [Cy]

**Addressing :** Register addressing

**Group :** Arithmetic group

**Bytes :** 1 byte

**Comment:** This instruction adds the content of register to the content of accumulator and the content of the carry flag. The result is placed in accumulator. All flags may be affected.

**Example:** Let [A] = 5F H, [D] = 33 H AND [Cy] = 01 H

**Instruction:** ADC D

**Addition:**
- [A] : 5F H = 0 1 0 1 1 1 1 1
- [D] : +33 H = 0 0 1 1 0 0 1 1
- [Cy] : +01 H = 0 0 0 0 0 0 0 1
- [A] = 93 H = 1 0 0 1 0 0 1 1

**Flags**
- S = 1, Z = 0, Ac = 0
- P = 1, Cy = 0

[Note: This instruction generally used in 16-bit addition. For e.g. to add the content of BC register to the content of DE registers, this instruction is used to account for the carry generated by low order byte]

5) **ADC M : [ADD MEMORY CONTENT TO ACCUMULATOR WITH CARRY]**

: [A] ← [A] + [[H-L]] + [Cy]

**Addressing :** Register Indirect

**Group :** Arithmetic group

**Bytes :** 1 byte

**Comment:** The content of memory location whose address place in H-L register pair and content of Cy flag are added to the content of accumulator. The result is placed in accumulator.

**Example:** Let [HL] = F000H

[A] = 35 H

[Cy] = 00H, [F000H] = 05 H
Instruction: ADC M
After execution: \([A] = 3\text{AH}\)

Flag =

<table>
<thead>
<tr>
<th>S</th>
<th>Z</th>
<th>Ac</th>
<th>P</th>
<th>Cy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

6) **ACI data** : [ADD IMMEDIATE TO ACCUMULATOR WITH CARRY]

: \([A] \leftarrow [A] + \text{data} + [\text{Cy}]\)

Addressing : Immediate addressing

Group : Arithmetic group

Bytes : 2 byte

**Comment:** This instruction adds the content of accumulator to the 8-bit immediate data specified in second byte of instruction along with the content of carry flag. The result is placed in accumulator. All flags may be affected.

Example : Let \([\text{Cy}] = 1\text{H}\) and \([A] = 05\text{H}\)

Instruction: ACI 55\text{H}

After execution: \([A] = 5\text{B}\text{H}\)

7) **SUB r** : [SUBTRACT REGISTER FROM ACCUMULATOR]

: \([A] \leftarrow [A] - [r]\)

Addressing : Register addressing

Group : Arithmetic instructions group

Bytes : 1 byte

**Comment:** The content of register \(r\) are subtracted from the content of accumulator. The result is placed in accumulator. All the flags may be affected.

Example

: \([A] = 37\text{H}\)

: \([C] = 40\text{H}\)

Instruction: SUB C

\([C] : 40\text{H} = 0100\ 0000\)

2’s complement = 1100 0000

+ \([A] : 37\text{H} = 0011\ 0111\)

\[\begin{array}{c}
0 \\
/ 1111 \ 0111
\end{array}\]

Complement carry

\[\begin{array}{c}
1 \\
/ 1111 \ 0111
\end{array}\]

Result: \([A] = F7\text{H}\)

Flags: \(S = 1, Z = 0, Ac = 0,\)

\(P = 0, Cy = 1\)

The result, as a negative number, will be in 2’s complement and thus the carry (Borrow) flag is set.

8) **SUB M** : [SUBTRACT MEMORY FROM ACCUMULATOR]

: \([A] \leftarrow [A] - [[H-L]]\)
Addressing : Register indirect
Group : Arithmetic instructions group
Bytes : 1 byte

**Comment:** The content of memory location, whose address stored in H-L register pair is subtracted from the content of accumulator. The result is placed in the accumulator.

Example : [HL] = 2500 H  
          [2500] = 05 H  
          [A] = 07 H  
          Instruction: SUB M  
          After execution: [A] = 02 H

9) **SUI data** : [SUBTRACT IMMEDIATE FROM ACCUMULATOR]
          : [A] ← [A] – data
Addressing : Immediate addressing
Group : Data transfer group
Bytes : 2 byte

**Comment:** The 8-bit immediate data specified in the second byte of the instruction is subtracted from the content of accumulator. Result is placed in accumulator. All the flags may be affected.

Example : Let, [A] = 1F H  
          Instruction: SUI 1FH  
          After execution: [A] = 00H

10) **SBB r** : [SUBTRACT REGISTER AND BORROW FROM ACCUMULATOR]
          : [A] ← [A] – [r] – [Cy]
Addressing : Register addressing
Group : Arithmetic group
Bytes : 1 byte

**Comment:** The contents of register r and carry bit are subtracted from the contents of accumulator. The result is placed in accumulator. All the flags may be affected.

Example : [A] = 37 H  
          [B] = 3F H  
          [Cy] = 01 H  
          Instruction: SBB B  
          (B): 3 F  
          Borrow: +1  
          
          \[
          \begin{array}{c}
          \text{40 H} = 0100 \ 0000 \\
          \text{2’ complement of 40 H} \\
          = 1100 \ 0000 \\
          \end{array}
          \]

          \[
          \begin{array}{c}
          + (A) = 0011 \ 0111 \\
          \hline
          0 \ / \ 1111 \ 0111 \\
          \end{array}
          \]
Complement carry:

\[ \begin{array}{c|c|c|c|c|c|c|c} & & & & & & & \\
\hline & & & & & & & \\
& & & & & & & \\
& & & & & & & \\
& & & & & & & \\
& & & & & & & \\
& & & & & & & \\
& & & & & & & \\
& & & & & & & \\
& & & & & & & \\
& & & & & & & \\
\end{array} \]

\[
\begin{array}{cccccccc}
1 & & & & & & & 1 1 1 1 0 1 1 1
\end{array}
\]

Result: \( [A] = F7H \)

The below flag is set to indicate the result’s is in 2’s complement.

11) **SBB M** : [SUBTRACT MEMORY CONTENT AND BORROW FROM ACCUMULATOR]

\[
[A] \leftarrow [A] - ([H][L]) - [Cy]
\]

Addressing : Register Indirect addressing

Group : Arithmetic group

Bytes : 1 byte

**Comment**: The content of memory location whose address is stored in H-L pair along with carry bit are subtracted from the contents of accumulator. Result is placed in accumulator. All the flags may be affected.

Example : Let \([H-L] = 2500 H, [2500] = 05 H, [A] = 07 H \text{ and } [Cy] = 0\)

Instruction: SBB M

After execution: \([A] = 02 H\)

12) **SBI data** : [SUBTRACT IMMEDIATE WITH BORROW]

\[
[A] \leftarrow [A] - DATA - [Cy]
\]

Addressing : Immediate addressing

Group : Arithmetic group

Bytes : 2 byte

**Comment**: The 8-bit immediate data, specified in the second byte of instruction is subtracted along with the carry bit from the content of accumulator. The result is placed in accumulator. All the flags may be affected.

Example : Let \([A] = 32 H, [Cy] = 1 H,\)

Instruction: SBI 31 H

After execution: \([A] = 0\)

13) **INR r** : [INCREMENT REGISTER CONTENT BY 1]

\[
[r] \leftarrow [r] + 1
\]

Addressing : Register addressing

Group : Arithmetic group

Bytes : 1 byte

**Comment**: The content of register \(r\) are incremented by one and the result are stored in the same place. All the flags except carry flag may be affected. The register \(r\) can be A, B, C, D, E, H and L.

Example : Let \([B] = FFH\)

Instruction: INR B

After execution: \([B] = 00H\)

14) **INR M** : [INCREMENT MEMORY CONTENT BY 1]

\[
[[H][L]] \leftarrow [[H]][[L]] + 1
\]

Addressing : Register indirect
Group: Arithmetic instruction
Bytes: 1 byte

**Comment:** The content of memory location whose address is stored in H-L register pair is incremented by one and result again store on the same place.

Example: 
[H-L] = 2500 H

* [2500] = 04 H
* Instruction: INR M
* After execution: [2500] = 05 H

15) **INX rp**

*: [INCREMENT REGISTER PAIR BY 1]*

: [rp] ← [rp] + 1

Addressing: Register addressing
Group: Arithmetic group
Bytes: 1 byte

**Comment:** The instruction increments the Memory Location of register pair rp by 1. No flags are affected.

The instruction views the contents of the two registers as a 16-bit number.

Example: 
Let [HL] = D000 H

* Instruction: INX H
* After execution: [HL] = D001 H

16) **DCR r**

*: [DECREMENT REGISTER PAIR BY 1]*

: [r] ← [r] - 1

Addressing: Register
Group: Arithmetic
Bytes: 1 byte

**Comment:** The content of register is decremented by 1 and the results are stored in the same place.

Example: 
[D] = 00 H

* Instruction: DCR D
* [D] = 00 H = 0 0 0 0 0 0 0 0
  - 1 H = 0 0 0 0 0 0 0 1

Subtraction is performed in 2’s complement

[D] = 0 0 0 0 0 0 0 0 +

2’s complement 1 1 1 1 1 1 1 1

of 1

[D] = 1 1 1 1 1 1 1 1

After execution: [D] FFH

17) **DCR M**

*: [DECREMENT MEMORY CONTENT BY 1]*

: [[H][L]] ← [[H][L]] - 1

Addressing: Register indirect addressing
**Group**: Arithmetic group  
**Bytes**: 1 byte  
**Comment**: This instruction decrements the content of memory location, whose address is stored in H-L pair by 1 and the result is placed at same place. All flags except carry flag are affected.

**Example**: Let [H-L] = D000H and [D000] = 2A = H  
**Instruction**: DCR M  
**After execution**: [D000] = 29H

18) **DCX rp**: [DECREMENT REGISTER PAIR BY 1]  
: [rp] ← [rp] - 1  
**Addressing**: Register addressing  
**Group**: Arithmetic group  
**Bytes**: 1 byte  
**Comment**: This instruction decrements the Memory Location of register pair rp by 1. No flags are affected. This instruction vies the contents of the two registers as a 16-bit number.

**Example**: Let [DE] = D000 H  
**Instruction**: DCX D  
**After execution**: [DE] = CFFF H

19) **DAD rp**: [ADD REGISTER PAIR TO H AND L REGISTER]  
: [H] [L] ← [H] [L] + [rh] [rl]  
**Addressing**: Register addressing  
**Group**: Arithmetic group  
**Bytes**: 1 byte  
**Comment**: The contents of register pair rp are added to the contents of H-L pair. Result is placed in register H and L. Only carry flag is affected.

**Instruction**: DAD D  
**After execution**: [L] = 05 + 12 = 17 H  
[H] = 03 + 15 = 18 H  
: [H-L] = 1817 H

In this case, carry flag is reset.

20) **DAA**: [DECIMAL ADJUST ACCUMULATION]  
**Addressing**: Implied addressing  
**Group**: Arithmetic group  
**Bytes**: 1 byte  
**Comment**: The eight bit number in the accumulator is adjusted to form two four—bit Binary-coded Decimal digits by this instruction. It can be done by following process:

1) If the value of the least significant 4 bits of the accumulator (A₃ – A₀) is greater than 9 or if the AC flag is set, 6 (06) is added to low order 4-bits of accumulator.
2) If the value of most significant 4-bits of the accumulator (A<sub>7</sub> – A<sub>4</sub>) is greater than 9 or if the Cy flag is set, 6 (60) is added to the high order 4-bits of accumulator.

3) If both 4 LSBs and 4 MSBs of accumulator are greater than 9 or Ac and C flags are set respectively then 66 add to the accumulator content.

[Note: This instruction must always follow an addition instruction for two BCD numbers. It can not be used to adjust results after subtraction.]

Example: Add 12 BCD to BCD

\[39_{BCD} = 0\ 0\ 1\ 1\ 1\ 0\ 0\ 1\]
\[+12_{BCD} = 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\]
\[51_{BCD} = 0\ 1\ 0\ 0\ 1\ 0\ 1\ 1 = 4BH\]

The binary sum is 4B H. But BCD sum is 51

To adjust result add 6 to lower nibble

\[4B = 0\ 1\ 0\ 0\ 1\ 0\ 1\ 1\]
\[+6B = 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\]
\[51 = 0\ 1\ 0\ 1\ 0\ 0\ 0\ 1\]

Thus [A] = 51 i.e. contents are adjusted to BCD values.

III Logical Group:

1) **ANA r**: [LOGICAL AND WITH ACCUMULATOR]

\[: [A] \leftarrow [A] \land [r]\]

Addressing: Register addressing

Group: Logical group

Bytes: 1 byte

Comment: The content of accumulator are logically ANDed with the content of register r. Result is placed in accumulator. S, Z and P flags are modified. The Cy flag is reset and Ac flag is set.


Instruction: ANA B

\[\text{[A]}: 25\ H = 0\ 0\ 1\ 0\ 0\ 1\ 0\ 1\]
\[\text{AND [B]}: 31\ H = 0\ 0\ 1\ 1\ 0\ 0\ 0\ 1\]
\[= 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ = 21\ H\]

After execution: [A] = 21 H

Flags: S = 0, Z = 0, P = 1,
Ac = 1, Cy = 0

2) **ANA M**: [LOGICAL AND WITH MEMORY]

\[: [A] \leftarrow [A] \land ([H] [L])\]

Addressing: Register indirect addressing

Group: Logical group

Bytes: 1 byte
**Comment:** The content of accumulator are logically ANDed with the content of memory location, whose address is stored in H-L pair. The result is placed in accumulator. The S,Z and P flags are modified. The Cy flag is reset and the Ac flag is set.

**Example:** Let \([A] = 3B\) H; \([H\text{-}L] = D000\) H and \([D000] = 29\) H

Instruction: **ANA M**

\[
[A]: 3B = 00111011 \\
\text{AND:} \ 29 = 00101001 \\
\text{Result:} \ 00101001 = 29\ H
\]

After execution: \([A] = 29\) H

Flags: S = 0, Z = 0, P = 0,

Ac = 1, Cy = 0

3) **ANI data** : [AND IMMEDIATE WITH ACCUMULATOR]

: \([A] \leftarrow [A] \land \text{data}\)

**Addressing** : Immediate addressing

**Group** : Logical group

**Bytes** : 2 byte

**Comment:** The content of accumulator are logically ANDed with the 8-bit immediate data specified in the second byte of the instruction. The result is placed in the accumulator. The S, Z, and P flags are modified. Cy flag is cleared and Ac flag is set.

**Example:** Let \([A] = 11\) H

Instruction: **ANI 11\ H**

After execution: \([A] = 11\) H

4) **ORA r** : [LOGICALLY OR WITH ACCUMULATOR]

: \([A] \leftarrow [A] \lor [r]\)

**Addressing** : Register addressing

**Group** : Logical group

**Bytes** : 1 byte

**Comment:** The content of accumulator are logically ORed with the content of register r. The result is placed of accumulator. r may be any one of A, B, C, D, E, H, A and L register. Ac and Cy flags are reset.

**Example:** Let \([A] = 29\) H and \([B] = 35\) H

Instruction: **ORA B**

\[
[A]: 29 = 00101001 \\
\text{OR 35 [B]:} \ H = 00110101 \\
\text{Result:} \ 00111101 = 3D\ H
\]

After execution: \([A] = 3D\) H

Flags: S = 0, Z = 0, P = 0, Ac = 0, Cy = 0

5) **ORA M** : [LOGICALLY OR WITH MEMORY]

: \([A] \leftarrow [A] \lor [[H]][[L]]\)
Addressing: Register Indirect
Group: Logical
Bytes: 1 byte

Comment: The content of memory location, whose address placed in H-L register pair are logically ORed with content accumulator. The result is placed in accumulator. Ac and Cy flags are reset.

Example: \([A] = 03 \text{H} \]
\([H-L] = \text{D}000 \text{H} \]
\([D000] = 81 \text{H} \]
Instruction: ORA M
\[ 03 \text{H} = 0 0 0 0 0 0 0 1 1 \]
\[ \text{OR } 81 \text{H} = 1 0 0 0 0 0 0 1 \]
\[ 83 \text{H} = 1 0 0 0 0 0 0 1 1 \]
After execution: \([A] = 83 \text{H} \]
Flags: S = 1, Z = 0, P = 0, Ac = 0, Cy = 0

6) ORI data: \([\text{LOGICALLY OR IMMEDIATE}]\)
\([A] \leftarrow [A] \lor \text{data} \]
Addressing: Immediate addressing
Group: Logical group
Bytes: 2 byte

Comment: The content of accumulator are logically ORed with the 8-bit immediate data specified in the second byte of the instruction. The result is placed in accumulator. The S, Z and P flags are affected. The Cy and Ac flags are reset.

Example: Let, \([A] = 35 \text{H} \]
Instruction: ORI 99H
\([A] 35 \text{H} = 0 0 1 0 0 1 0 1 \]
\[ \text{OR } 99 \text{H} = 1 0 0 1 1 0 0 1 \]
\[ 1 0 1 1 1 1 0 1 = \text{BD} \text{H} \]
After execution: \([A] = \text{BDH} \]
Flags: S = 1, Z = 0, P = 1, Ac = 0, Cy = 0

7) XRA r: \([\text{EXCLUSIVE OR WITH ACCUMULATOR}]\)
\([A] \leftarrow [A] \oplus r \]
Addressing: Register addressing
Group: Logical group
Bytes: 1 byte

Comment: The content of register are logically exclusive ORed with the content of accumulator. The result is placed in accumulator. The r may be any one of the A, B, C, D, E, and L register. The Cy and Ac flags are reset.
Instruction: XRA B
[A] 25 H = 0 0 1 0 0 1 0 1
[B] 39 H = 0 0 1 1 1 0 0 1

After execution: [A] = 1C H
Flags: S = 0, Z = 0, P = 0, Ac = 0, Cy = 0

8) XRA M : [EXCLUSIVE OR WITH MEMORY]
    : [A] ← [A] ⊕ [[H-L]]
Addressing : Register Indirect
Group : Logical group
Bytes : 1 byte
Comment: The content of the of the memory location whose address placed in H-L register pair are logically exclusive Ored with the content accumulator. The result is placed in the accumulator. The Cy and Ac flags are reset.
Example : Let, [A] = 77 H
[H-L] = D000H
[D000] = 56 H
Instruction: XRA M
[A] 77 H = 0 1 1 0 0 1 1 1
[D000] 56 H = 0 1 0 1 0 1 1 0

After execution: [A] = 21 H
Flags: S = 0, Z = 0, P = 1, Ac = 0, Cy = 0

9) XRI data : [EXCLUSIVE OR IMMEDIATE WITH ACCUMULATOR]
    : [A] ← [A] ⊕ data
Addressing : Immediate addressing
Group : Logical group
Bytes : 2 byte
Comment: The content of accumulator are logically exclusive. ORed with the 8-bit immediate data specified in second byte of instruction. The result is placed in accumulator. The S,Z and P flags are affected. The Cy and Ac flags are reset.
Example : Let, [A] = 5B H
Instruction: XRI 35 H
[A] : 5B H = 0 1 0 1 1 0 1 1
Data 35 H = 0 0 1 1 0 1 0 1

After execution: [A] = 6E H
10) **CMR r** : [COMPARE WITH ACCUMULATOR]

: \([A] \rightarrow [r]\]

**Addressing**: Register addressing

**Group**: Logical group

**Bytes**: 1 byte

**Comment**: This instruction compares the content of the register with content of accumulator. Comparison is done using subtraction of content of register from the content of accumulator. The content of accumulator remains unchanged.

The result of comparison is shown by setting the flags as:

(a) if \([A] < [r]\) then Cy flag is set to 1 & Z flag is RESET (0)

(b) if \([A] = [r]\) then Z flag is set to 1 Cy flag is SET

(c) if \([A] > [r]\) then Z and Cy flags are reset.

The ‘r’ may be any one of the A, B, C, D, E, H and L register.

**Example**: Let \([A] = 15\) H and \([H] = 57\) H

Instruction: CMP H

After execution: Cy = 1, Z = 0.

11) **CMP M** : [COMPARE MEMORY WITH ACCUMULATOR]

: \([A] \rightarrow [[[H-L]]]\]

**Addressing**: Register indirect addressing

**Group**: Logical group

**Bytes**: 1 byte

**Comment**: This instruction compares the content of memory location whose address is stored in H-L pair with the content of accumulator by subtracting the content of memory location from the content of accumulator. The content of accumulator remains unchanged.

The result of comparison is shown by setting the flags as below:

(a) The zero flag is set to 1 if \([A] = [[[H-L]]]\]

(b) The Cy flag is set to 1 if \([A] < [[[H-L]]]\]

(c) Both Cy and Z flags are reset if \([A] > [[[H-L]]]\]

12) **CPI data** : [COMPARE IMMEDIATE WITH ACCUMULATOR]

: \([A] \rightarrow \text{data}\)

**Addressing**: Immediate addressing

**Group**: Logical group

**Bytes**: 2 byte

**Comment**: This instruction compares the 8-bit immediate data, specified in the second byte of instruction, by subtracting it from the contents of accumulator. The content of accumulator remains unchanged. The results of comparison is shown by setting flags as:

(1) If the content of accumulator are equal to 8-bit immediate data, then zero flag is set.
(2) If the content of accumulator are less than the 8-bit immediate data, then carry flag is set.
(3) Else, both flags are reset.

13) **RLC**
*: [ROTATE ACCUMULATOR LEFT]*

*: [A_{n+1}] \leftarrow [A_n], [A_0] \leftarrow [A_7], [C_y] \leftarrow [A_7]*

**Addressing**: Implied addressing

**Group**: Logical group

**Bytes**: 1 byte

**Comment**: The content of a accumulator are rotated to left by one bit position. The bit A, is stored in bit A_0, as well as in carry flag. It is shown in following figure:

![RLC Diagram]

**Example**: Let \([A] = 93 \text{ H} \), \([C_y] = 0\)

Instruction: RLC

Before instruction:

<table>
<thead>
<tr>
<th>A_7</th>
<th>A_0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1 0 0 1 0 0 1 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cy</th>
<th>Accumulator</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-------------</td>
</tr>
</tbody>
</table>

After instruction: RLC

<table>
<thead>
<tr>
<th>A_7</th>
<th>A_0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0 0 0 0 0 1 1 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cy</th>
<th>Accumulator</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-------------</td>
</tr>
</tbody>
</table>

Thus \([A] = 27\) and \(C_y = 1\)

14) **RRC**
*: [ROTATE ACCUMULATOR RIGHT]*

*: [A_0] \leftarrow [A_{n+1}], [A_7] \leftarrow [A_0], [C_y] \leftarrow [A_0]*

**Addressing**: Implied addressing

**Group**: Logical group

**Bytes**: 1 byte

**Comment**: The content of accumulator are rotated right by one position. The bit A_0 of accumulator is stored in the bit A_7 as well as in carry flag. Only the C_y flag is affected. The function of RRC is shown in following figure:

![RRC Diagram]

**Example**: Let \([A] = 83 \text{ H} \), \([C_y] = 0\)

<table>
<thead>
<tr>
<th>A_7</th>
<th>A_0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Cy
Accumulator

Instruction: RRC

After execution:

Cy
Accumulator

Thus \([A] = 27\) and \(Cy = 1\)

15) **RAL**

: [ROTATE ACCUMULATOR LEFT THROUGH CARRY]

: \([A_{n+1}] \leftarrow [A_n], [A_0] \leftarrow [Cy], [Cy] \leftarrow [A_7]\)

**Addressing**: Implied addressing

**Group**: Logical group

**Bytes**: 1 byte

**Comment**: The instruction rotates the content of accumulator one position left through carry flag. The carry flag status is stored in bit \(A_0\) of accumulator and the bit \(A_7\) of accumulator is stored in carry flag.

The function of RAL is shown in following figure:

![Diagram of RAL function]

Example: Let \([A] = 29\) H

\([Cy] = 1\) H

Cy
Accumulator

Instruction: RAL

Cy
Accumulator

Thus \([A] = 53\) H

\(Cy = 0\)

16) **RAR**

: [ROTATE ACCUMULATOR RIGHT THROUGH CARRY]

: \([A_n] \leftarrow [A_{n+1}], [A_7] \leftarrow [Cy], [Cy] \leftarrow [A_0]\)

**Addressing**: Implied addressing

**Group**: Logical group

**Bytes**: 1 byte

**Comment**: The content of the accumulator are rotated to right by one bit position through carry flag. The carry flag status is stored in bit \(A_7\) of accumulator and the bit \(A_0\) of accumulator is stored in carry flag. Only the carry flag is affected.

The function of RAR is shown in following figure:
Example : Let \([A] = 29 \text{ H}\)

\[[Cy] = 1 \text{ H}\]

<table>
<thead>
<tr>
<th>Cy</th>
<th>Accumulator</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 0 1 0 1 0 0 1</td>
</tr>
</tbody>
</table>

Instruction: RAR

<table>
<thead>
<tr>
<th>Cy</th>
<th>Accumulator</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0 0 0 1 1 1 0 1</td>
</tr>
</tbody>
</table>

Thus \([A] = 1D \text{ H}\)

\([Cy] = 1\)

17) **CMA** : [COMPLEMENT THE ACCUMULATOR]

\([A] \leftarrow [\overline{A}]\)

Addressing : Implied addressing

Group : Logical group

Bytes : 1 byte

**Comment**: This Instruction complements the content of accumulator. Result is placed in accumulator.

Example : Let \([A] = 3B \text{ H} = 0011011\)

Instruction: CMA

After execution: \([A] = 1100100\)

i.e. \([A] = C4 \text{ H}\)

18) **CMC** : [COMPLEMENT CARRY]

\([Cy] \leftarrow [\overline{Cy}]\)

Group : Logical group

Bytes : 1 byte

**Comment**: The carry flag is complemented. No other flags are affected.

Example : Let \([Cy] = 1 \text{ H}\)

Instruction: CMC

After execution: \([Cy] = 0 \text{ H}\)

19) **STC** : [SET CARRY]

\([Cy] \leftarrow 1\)

Addressing : Implied addressing

Group : Logical group

Bytes : 1 byte

**Comment**: This instruction sets carry flag to 1. No other flags are affected.
IV  **Branching Group:**

1) **JMP addr.** : [JUMP UNCONDITIONALLY]
   
   : [PC] ← addr

   **Addressing**: Immediate addressing
   **Group**: Branching group
   **Bytes**: 3 byte

   **Comment**: The control is transferred unconditionally to the memory location, whose address is specified in the instruction.

2) **Jcondition addr.** : [CONDITIONAL JUMP]
   
   : [PC] ← addr

   **Addressing**: Immediate addressing
   **Group**: Branching group
   **Bytes**: 3 byte

   In conditional jump instruction the jump is taken only if the condition is true. The conditional jump instructions and conditions are given below.
   
   i) **JNZ addr**: Jump on not zero (z = 0)
   
   ii) **JZ addr**: Jump on zero (z = 1)
   
   iii) **JNC addr**: Jump on not carry (Cy = 0)
   
   iv) **JC addr**: Jump on carry (Cy = 1)
   
   v) **JPO addr**: Jump on odd parity (P = 0)
   
   vi) **JPE addr**: Jump on even parity (P = 1)
   
   vii) **JP addr**: Jump on plus (S = 0)
   
   viii) **JM addr**: Jump on minus (S = 1)

   If the condition is satisfied, then only the address of memory location specified in the instruction is loaded in program counter.

3) **CALL addr.** : [UNCONDITIONAL SUBROUTINE CALL]

   : [SP] ← [PCu]
   
   : [[SP] ← [PCu]]
   
   : [[SP] ← [PCu]]
   
   : [SP] ← [SP] - 2
   
   : [PC] ← addr

   **Addressing**: Immediate addressing
   **Group**: Branching group
   **Bytes**: 3 byte

   **Comment**: CALL instruction is used to call a subroutine unconditionally. Before the control is transferred to the subroutine, the address of next instruction to be executed of the main program is stored in the stack. The content of SP are decremented by 2. Then, the program jumps to the subroutine whose starting address is specified in the instruction.
4) **Condition addr.** : [CONDITIONAL CALL]

\[
\begin{align*}
([SP] - 1) & \leftarrow [PC_1] \\
([SP] - 2) & \leftarrow [PC_1] \\
[SP] & \leftarrow [SP] - 2 \\
[PC] & \leftarrow \text{addr}
\end{align*}
\]

The conditional call instruction and conditions are listed below:

i) **CC addr** : Call if carry \((Cy = 1)\)

ii) **CNC addr** : Call if no carry \((Cy = 0)\)

iii) **CZ addr** : Call if zero \((Z = 1)\)

iv) **CNZ addr** : Call if no zero \((Z = 0)\)

v) **CP addr** : Call if plus \((S = 0)\)

vi) **CM addr** : Call if minus \((S = 1)\)

vii) **CPO addr** : Call if odd parity \((P = 0)\)

viii) **CPE addr** : Call if even parity \((P = 1)\)

5) **RET** : [RETURN FROM SUBROUTINE]

\[
\begin{align*}
[PC_1] & \leftarrow [SP]. \\
[PC_0] & \leftarrow [SP] + 1 \\
[SP] & \leftarrow [SP] + 2
\end{align*}
\]

Addressing : Register Indirect

Group : Branching group

Bytes : 1 byte

Comment: The content of memory location, whose address is specified in stack pointer are moved to the lower order byte of program counter. The content of the memory location whose address is one more than the content of SP, moved to the higher order byte of program counter. The content of stack pointer are incremented by 2.

6) **Rcondition** : [CONDITIONAL RETURN]

\[
\begin{align*}
[PC_1] & \leftarrow [SP], \\
[PC_0] & \leftarrow [SP] + 1 \\
[SP] & \leftarrow [SP] + 2
\end{align*}
\]

Addressing : Register Indirect

Bytes : 1 byte

Comment: If the specified condition is true, the action specified in RET are performed. Otherwise the control continues sequentially.

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Description</th>
<th>Flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC</td>
<td>Return on Carry</td>
<td>Cy = 1</td>
</tr>
</tbody>
</table>
RNC  Return with no carry  Cy = 0
RP   Return on positive  S = 0
RM   Return on minus  S = 1
RPE  Return on parity even  P = 1
RPO  Return on parity odd  P = 0
RZ   Return on zero  Z = 1
RNZ  Return on no zero  Z = 0

7) **RST n : [RESTART]**

\[ ([SP] - 1) \leftarrow [PC_n], \]
\[ ([SP] - 2) \leftarrow [PC_L], \]
\[ [SP] \leftarrow [SP] - 2 \]
\[ [PC] \leftarrow 8 \times (n) \]

Addressing  : Register Indirect
Bytes  : 1 byte

**Comment:** Control is transferred to the instruction whose address is 8 times the content of n. This instruction are used with interrupts.

**Opcode/operand**  | **Description**
---|---
RST 0  | 0000
RST 1  | 0008
RST 2  | 0010
RST 3  | 0018
RST 4  | 0020
RST 5  | 0028
RST 6  | 0030
RST 7  | 0038

8) **PCHL : [LOAD PROGRAM COUNTER WITH HL]**

\[ [PC_{HL}] \leftarrow [H] \]
\[ [PC_L] \leftarrow [L] \]

Addressing  : Register Indirect
Group  : Branching group
Bytes  : 1 byte

**Comment:** This instruction moves the content of register H to higher order byte of program counter and the content of register L to lower order byte of program counter.

This instruction is equivalent to one byte unconditional jump instruction, with jump address stored in H-L pair.

**Example:** Let [H] = 25 H and [L] = 39 H

Instruction: PCHL
After execution: \([PC] = 2539\) H

After execution of PCHL instruction, the control will be transferred to memory location 2539 H.

### Machine Control Group:

#### A) Stack operation:

1) **PUSH rp** : \([\text{PUSH REGISTER PAIR ON STACK}]\)

   \[
   \begin{align*}
   ([SP] - 1) & \leftarrow [rh] \\
   ([SP] - 2) & \leftarrow [rl] \\
   [SP] & \leftarrow [SP] - 2
   \end{align*}
   \]

   **Addressing** : Register indirect addressing

   **Bytes** : 1 byte

   **Comment:**
   - (a) The content of the higher order register of register pair rp are moved to memory location, whose address is one less than the content of stack pointer.
   - (b) The content of the low order register of register pair rp area moved to the memory location whose address is two less than the content of stack pointer.
   - (c) The stack pointer is determined by two rp may be any one of the B (B & C), D (D & E), H (H & L).

   **Example** : Let \([SP] = D015\) H \([B] = 25\) H and \([C] = 55\) H

   **Instruction:** PUSH B

   **After execution:**

   \[
   \begin{align*}
   [D014] & = 25\ H \\
   [D013] & = 55\ H \\
   \text{And} & \ [SP] = D013\ H
   \end{align*}
   \]

   **Stack**

   \[
   \begin{array}{ccc}
   \text{SP} & D013 & 55 \\
   D014 & 25 & \\
   D015 & X & \\
   \end{array}
   \]

2) **PUSH PSW** : \([\text{PUSH ACCUMULATOR AND FLAG REGISTER ON STACK}]\)

   \[
   \begin{align*}
   ([SP] - 1) & \leftarrow [A] \\
   ([SP] - 2)_0 & \leftarrow [Cy], & ([SP] - 2)_1 & \leftarrow x, \\
   ([SP] - 2)_2 & \leftarrow [P], & ([SP] - 2)_3 & \leftarrow x, \\
   ([SP] - 2)_4 & \leftarrow [Ac], & ([SP] - 2)_5 & \leftarrow x, \\
   ([SP] - 2)_6 & \leftarrow [Z], & ([SP] - 2)_7 & \leftarrow [S], \\
   [SP] & \leftarrow [SP] - 2 & (X – \text{Undefined})
   \end{align*}
   \]
Addressing : Register indirect addressing
Bytes : 1 byte

Comment: (a) The content of accumulator are moved to the memory location, whose address is one less than the content of stack pointer.
(b) The content of processor status word (flag register) are moved to the memory location, whose address is two less than the content of stack pointer.
(c) The stack pointer is decremented by 2.

Example : Let \([A] = 33 \text{H} \) and Flag Register = 25 \text{H}, \([SP] = D015\)
Instruction: PUSH PSW
After execution: \([D014] = 33 \text{H}, [D013] = 25 \text{H}\)
\([SP] = D013 \text{H}\)

3) POP rp : [POP OFF STACK TO REGISTER PAIR]
   \([rl] \leftarrow [SP]\]
   \([rh] \leftarrow [SP + 1]\]
   \([SP] \leftarrow [SP + 2]\]
Addressing : Register indirect
Bytes : 1 byte

Comment: (a) The content of the memory location, whose address is specified by the stack pointer are moved to low order register of register pair rp.
(b) The content of the memory location, whose address is one more than the content of stack pointer are moved to high order register of register pair rp.
(c) The stack pointer is incremented by 2.
Rp may be any one of the pair B (B & C), D (D & E) and H (H & L)
Example : Let \([SP] = 2001 \text{H}\)
Instruction: POP H

Before Execution          After execution

\[
\begin{array}{c|c|c}
\hline
\text{Stack} & \text{Stack} \\
\hline
2001 & 2001 \\
2002 & 2002 \\
2003 & 2003 \\
\hline
\end{array}
\]
\([SP] = 2001\)          \([SP] = 2003\)
\([H] = 20 \text{H} [L] = 10 \text{H}\)

4) POP PSW : [POP OFF STACK TO ACCUMULATOR AND FLAG REGISTER]
   \([Cy] \leftarrow [[SP]]_0\]
   \([P] \leftarrow [[SP]]_2, [Ac] \leftarrow [[SP]]_4\]
   \([Z] \leftarrow [[SP]]_6, [S] \leftarrow [[SP]]_7\]
\[ [A] \leftarrow [[SP]] + 1 \]
\[ [SP] \leftarrow [SP] + 2 \]

**Addressing**: Register indirect

**Bytes**: 1 byte

**Comment**: (a) The content of the memory location, whose address is specified by the content of register stack pointer are used to restore the conditions flags.

(b) The content of memory location, whose address is one more than stack pointer are moved to accumulator.

(c) The content of stack pointer are incremented by 2.

5) **XTHL**: [EXCHANGE H AND L WITH TOP OF STACK]

\[ [L] \leftrightarrow [SP] \]
\[ [H] \leftrightarrow [[SP]] + 1 \]

**Addressing**: Register indirect

**Bytes**: 1 byte

**Comment**: The content of the L register are exchanged with the content of the memory location whose address is stored in stack pointer. The content of the H register are exchanged with the content of the memory location, whose address is one more than the content of the stack pointer content of SP are not altered.

**Example**: Let \([H] = 20\) H, \([L] = FF\) H

```
Instruction: XTHL

[H] = CDH [L] = ABH
```

6) **SPHL**: [MOVE HL TO SP]

\[ [SP_1] \leftarrow [L] \]
\[ [SP_2] \leftarrow [H] \]

**Addressing**: Register addressing

**Group**: Machine control group [stack operation]

**Bytes**: 1 byte

**Comment**: This instruction copies the content of register L into lower byte of stack pointer and the content of register H into higher order byte stack pointer. The content of register H and L are not affected. This instruction is used for initializing the stack pointer.

**Example**: Let, \([H] = 25\) H and \([L] = 59\) H

```
Instruction: SPHL

After execution: [SP] = 2559 H
```
B) **Other Instructions:**

1) **IN port**: [INOUT 8-BIT DATA FROM AN INPUT PORT TO ACCUMULATOR]
   
   - Instruction: [INOUT 8-BIT DATA FROM AN INPUT PORT TO ACCUMULATOR]
   - Addressing: Direct addressing
   - Group: Machine (I/O) control group
   - Bytes: 2 byte
   - **Comment**: When this instruction is executed, microprocessor sends 8-bit port address on lower order address bus i.e. A₀ to A₇. Then, the 8-bit data placed on the 8-bit bidirectional data bus by the specified port is moved to accumulator.

   **Example**: IN 10 H

   When this instruction is executed 8-bit data is inputed from a port, whose address is 10 H.

2) **OUT port**: [OUTPUT 8-BIT DATA FROM ACCUMULATOR TO AN OUTPUT PORT]
   
   - Instruction: [OUTPUT 8-BIT DATA FROM ACCUMULATOR TO AN OUTPUT PORT]
   - Addressing: Direct addressing
   - Group: Machine (I/O) control group
   - Bytes: 2 byte
   - **Comment**: When this instruction is executed, microprocessor sends 8-bit port address on the lower order address bus AD₀ to AD₇ 8-bit data is then transferred from accumulator to selected port.

   **Example**: OUT 32 H

   When this instruction is executed, microprocessor sends 8-bit data in accumulator to the port, whose address is 32 H.

3) **EI**: [ENABLE INTERRUPT]
   
   - Group: Machine control group
   - Bytes: 1 byte
   - **Comment**: EI means, Interrupt Enable. The interrupt system is enabled following the execution of the instruction next to EI and all interrupts are enabled.

4) **DI**: [DISABLE INTERRUPT]
   
   - Group: Machine control group
   - Bytes: 1 byte
   - **Comment**: DI means, disable interrupts. As soon as DI instruction is executed, the interrupt, system is disabled. Interrupts are not recognized during the DI instruction.

5) **HLT**: [HALT AND ENTER WAIT STATE]
   
   - Group: Machine control group
   - Bytes: 1 byte
   - **Comment**: When HLT instruction is executed, the processor is stopped. The register and flags and unaffected. This instruction is used to stop MPU. It is waiting for a peripheral device to finish its task and interrupt the processor. This is generally the last instruction of our assembly language program. An interrupt or reset is necessary to exit from Halt state.
6) **NOP** : [NO OPERATION]
Group : Machine control group
Bytes : 1 byte

**Comment:** When this instruction is executed, no operation is performed, only this instruction fetched and decoded. This instruction do not affect flags or content of registers. This instruction is useful to produce a time delay in a timing loop.

7) **RIM** : [READ INTERRUPT MASK]
Group : Machine control group
Bytes : 1 byte

**Comment:** This is a multipurpose instruction used to read the status of interrupts 7.5, 6.5, 5.5 and read serial data input bit. The instruction loads eight bits in the accumulator with the following interpretations:

![Diagram of RIM](image)

**Example:** After the execution of instruction RIM, the accumulator contained 49 H. Explain the accumulator contents:

[Schematic diagram]

8) **SIM** : [SET INTERRUPT MASK]
Group : Machine control group
Bytes : 1 byte

**Comment:** This is a multipurpose instruction and used to implement the 8085 interrupts (RST 7.5, 6.5 and 5.5) and serial data output.

The instruction interrupts the accumulator contents as follows:
**SOD:** Serial Output Data: Bit D_{7} of the accumulator is latched into the SOD output line and made available to a serial peripheral if bit D_{6} = 1.

**SDE:** Serial Output Enable: If this bit = 1, it enables the serial output. To implement serial output, this bit needs to be enabled.

**XXX:** Don’t Care

**R7.5:** Reset RST 7.5: If this bit = 1, RST 7.5 flip flop is reset. This is an additional control to reset RST 7.5

**MSE:** Mask Set Enable: If this bit is high, it enables the functions of bits D_{2}, D_{1}, D_{0}. This is a master control over all the interrupt masking bits. If this bit is low, bits D_{2}, D_{1}, D_{0} do not have any effect on the masks.

**M7.5:** D_{2} = 0, RST 7.5 is enabled
= 1, RST 7.5 is masked or disabled.

**M6.5:** D_{1} = 0, RST 6.5 is enabled
= 1, RST 6.5 is masked or disabled.

**M5.5:** D_{0} = 0, RST 5.5 is enabled
= 1, RST 5.5 is masked or disabled.