

ANL :

EX1:

45H 🡪 0100 0101

0FH 🡪 0000 1111

05H 0000 0101

EX2:

45H 🡪 0100 0101

F0H 🡪 1111 0000

40H 🡪 0100 0000

Note:AND logic is used to extract the nibbles of an hexadecimal byte.

|  |  |  |
| --- | --- | --- |
| NUMBER | ASCII | |
| DECIMAL | HEX |
| 0 | 48 | 30H |
| 1 | 49 | 31H |
| 2 | 50 | 32H |
| 3 | 51 | 33H |
| 4 | 52 | 34H |
| 5 | 53 | 35H |
| 6 | 54 | 36H |
| 7 | 55 | 37H |
| 8 | 56 | 38H |
| 9 | 57 | 39H |

ORL



04H 🡪 0000 0100

30H 🡪 0011 0000

34H 🡪 0011 0100

**Note :OR logic is useful to convert integer to ascii**.

XRL:



55H 🡪 0101 0101

FFH 🡪 1111 1111

AAH 🡪 1010 1010

**Note:**

1. **XOR logic is useful to compliment the byte.**
2. **Number XOR with same number then result is zero.**
3. **Number XOR with zero then result is same number.**

CPL : COMPLIMENT



54H 🡪 0101 0100 After executing CPL instruction result in A is:

ABH 🡪 1010 1011

RL: Rotate left

**MOV A,#89H;**

7 6 5 4 3 2 1 0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

**A**

**RL A**

7 6 5 4 3 2 1 0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |

**After RL instruction is executed the value in A is :**

7 6 5 4 3 2 1 0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |

**A**

**RR :Rotate right**

**MOV A,#99H**

7 6 5 4 3 2 1 0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |

**A**

**RR A**

7 6 5 4 3 2 1 0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |

**A**

**After rotating towards right the value in A is**

7 6 5 4 3 2 1 0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |

A

**Note:**

**In RL and RR instructions no flags are effected.**

**RLC (Rotate left through carry)**

**MOV A,#97H**

7 6 5 4 3 2 1 0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |

**A**

**RLC A**

7 6 5 4 3 2 1 0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |

**C**

The **RLC** instruction rotates the eight bits in the accumulator and the one bit in the carry flag left one bit position. Bit 7 of the accumulator is rotated into the carry flag while the original value of the carry flag is rotated into bit 0 of the accumulator. Bit 0 of the accumulator is rotated into bit 1, bit 1 into bit 2, and so on. No other flags are affected by this operation.

**Explaination:**

|  |
| --- |
| 0 |

Carry flag(c) is zero by default

C

7 6 5 4 3 2 1 0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |

A P=1;

C

7 6 5 4 3 2 1 0 P=0;

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |

|  |
| --- |
| 1 |

A C

7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |

DPH DPL