## Bitwise Operators :-

    &   ( bitwise and )
    |   ( or )
    ^   ( xor )
    ~   (      compliment )

```
int a=5,b=6,c,d,e,f;

c = a & b ;
d = a | b ;
e = a ^ b ;
f = ~a ;
```

1) c = a & b

```
    101
    110
    -----
    100   => 4
```

> If both bits are 1 then result is 1 otherwise 0.

2) d = a | b ;

```
    101
    110
    -----
    111   => 7
```

> If any one of the bit is 1 then result is 1 otherwise 0

3) e = a ^ b ;

```
    101
    110
    -----
    011   => 3
```

> XOR:-
> Opposite bits implies 1 , equal bits implies 0.

---

## Bitwise Operators :-

    &   ( bitwise and )
    |   ( or )
    ^   ( xor )
    ~   ( 1's compliment )

```
int a=5,b=6,c,d,e,f;

c = a & b ;
d = a | b ;
e = a ^ b ;
f = ~a ;
```

4) f = ~a;

> ~ makes all 0s to 1s and all 1s to 0s.

```
    101
    ~
    -----
    010
```

Value in a

```
0 | 000 0000 0000 0101
```

~

```
1 | 111 1111 1111 1010
```

-ve      It is 6 after 2's
number    compliment.

-6

For 2's complement → reverse each bit and add 1 to it

## Shift Operators :-

### 1) Left Shift ( << ) :-

Shifts given n bits to the left. Left most n bits will be dropped.

E.g.
    int a = 20;

Memory image of a is

    a   0 | 000 0000 0001 0100

Now

    int b = ( a << 4 );

Memory image of b is

    b   0 | 000 0001 0100 0000

Value of b is 320.

> If we shift more number of bits it may possible that '1' comes at $1^{st}$ place giving the negative number.

## Shift Operators :-

### 2) Right Shift ( >> ) :-

Operation of this operator is compiler dependent. It will shift given value n bits to the right. The newly formed left most bits are filled with zero in some compilers and with previous sign bit in some other compilers.
Standard is to fill them with previous sign bit value.

E.g.
    int a = 20;

Memory image of a is

    a   0 | 000 0000 0001 0100

Now

    int b = ( a >> 4 );

Memory image of b is

    b   0 | 000 0000 0000 0001

                    1

## Assignment Operators :-

Variable = Value ;

a = b + c;
②   ①

Special Property of assignment operator :-

int   a , b , c;

a   =   b   =   c   =   5;

5

5

5

Associtivity of assignment operator is Right to left and for '+' operator it is Left to Right

07:22                                                                                        11:08

## Arithmetic Assignment :-

+= , -= , /= , *= etc

Example :-

int a = 10;
a *= 2 ;

after which a will be 20.

Example 2:-

int a = 10;
a *= 2 + 2;

after which a is 40.
Since + operator has
more priority than *=
operator.

$$a \overset{2}{\circledast} = 2 \overset{1}{\oplus} 2$$

4

=> a = 40

$$a \overset{3}{=} a \overset{1}{\circledast} 2 \overset{2}{\oplus} 2$$

20

22

10:56                                                                                        11:08