Bitwise Operator

- The Bitwise operator are AND, OR, XOR, Complement, Left shift , Right Shift.
- These operators work on integral type of data and they perform operation on Binary representation of data I.e, 0's and 1's.
- Bitwise operators are used in applications of networking , Encryption and Decryption etc.
- Bitwise calculations starts from right hand side.

:

- To understand bitwise we first need to understand binary number system . Binary number system uses only 0's and 1's
- Decimal number system uses numbers from 0 - 9
- To convert decimal to binary we can do it in either 2 ways

Table for decimal to binary

- 1 * 1 = 1
  1 * 0 = 0
  0 * 1 = 0
  0 * 0 = 0

Ex : a - 1010

b - 1101

---

a & b - 1000 = 8 (Decimal form of binary number)

- OR works on addition
- Working of OR - 1 + 1 = 1
  1 + 0 = 1
  0 + 1 = 1
  0 + 0 = 1

Ex : a - 1010
b - 1101

---

a | b - 1111 = 15 (decimal form of binary number)

- Working of XOR - 1 ^ 1 = 0
  1 ^ 0 = 1
  1 ^ 1 = 1
  1 ^ 0 = 0

Ex: a - 1010
b - 1101

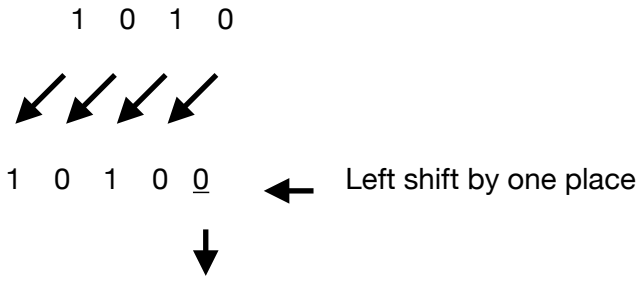
---

a ^ b - 0111 = 7 (decimal form of binary number)

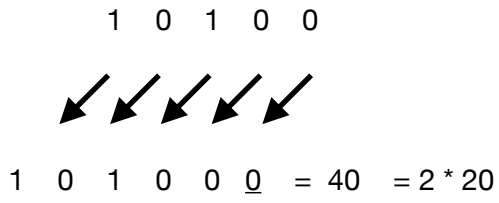
- Leading zero doesn't make sense
- Hence , we consider 111 whose decimal form is 7

Left shift (<<) and right shift (>>)

- a = 10 (binary of 10 = 1010)
- If we left shift ' a ' by one place I.e, a << 1 then



- After left shift by one place the bit that is freed will be taken as zero.
- Now 10100 = 20 = 2* 10
- If we left shift 20 again then,



- We see that when we left shift the number will double for one place I.e; a << n , a * 2 (power n)
- If we double the left shift I.e , a. << 2 then

  a << 2 = 2(power n) * a = 4*10 = 40

- similarly , if we do a << 5 , then the value gets double for 5 times

  a << 5 = 2(power 5) * a = 32 * 10 = 320

- If we RIGHT SHIFT the number will become half.

a = 10 -> 1 0 1 0

__ 1 0 1 0 (this 0 gets discarded)

This leading empty space has no value (it becomes zero)

- Right shift operator divide by 2 I.e; a >> n a / 2 (power n)