BITWISE OPERATORS

Bitwise operator's works on bits.

Turbo-c is a 16 bit compiler. Due to this bitwise operations are limited to 16 bits only [2^0 to 2^{15}].

Bitwise operators operate **integer** type values only.

We have to calculate only the **on** bits [**1**].

When the first bit[**Sign bit**] is **1** then the number is **Negative** and it is **0** then the number is **positive**.

They are very much used in system software development.

Note: Bitwise operator is low level feature.

C-Language supports following bitwise operators.

& -Bitwise and

| - Bitwise or

^ - XOR ==> Exclusive OR

~ - Compliment operator

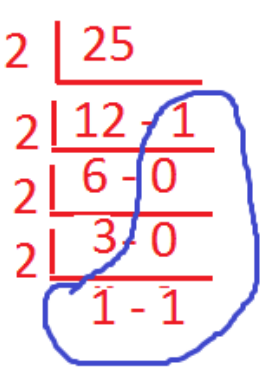
<< - Left shift operator

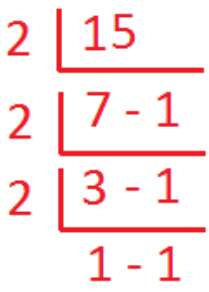
>> - Right shift operator

& - Bitwise and: In this both bits are 1's then result bit is 1. Otherwise result bit is 0.

Eg: **25 & 15 = 9**

25 = 0000 0000 0001 1001
15 = 0000 0000 0000 1111





25 & 15 = 9

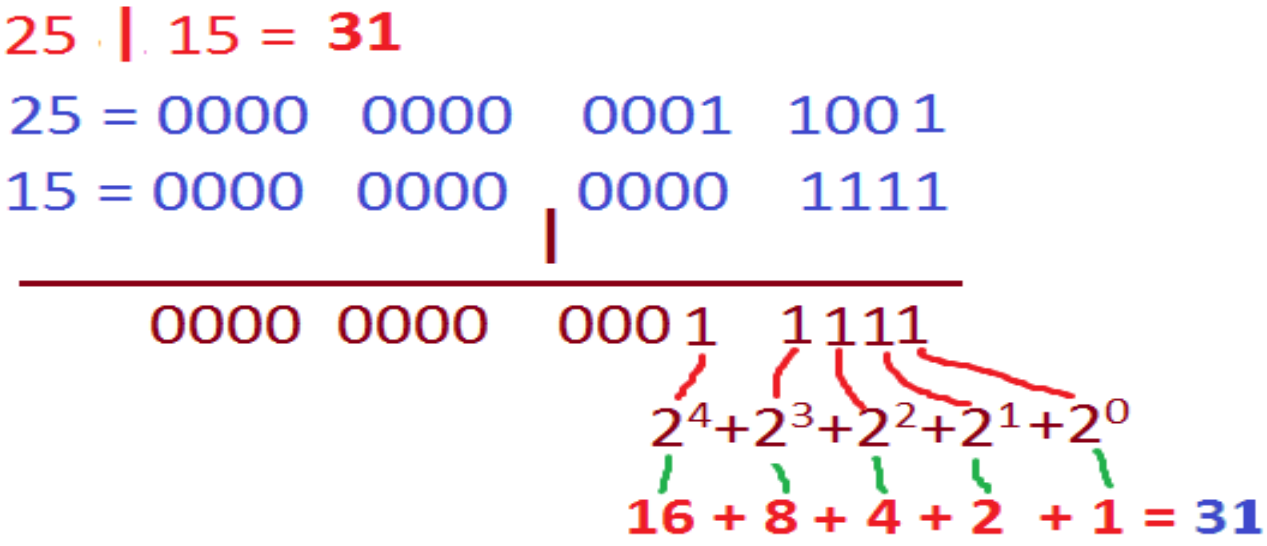
25 = 0000 0000 0001 1001
15 = 0000 0000 0000 1111
&
0000 0000 0000 1001

$2^3 + 2^0$

8 + 1 = **9**

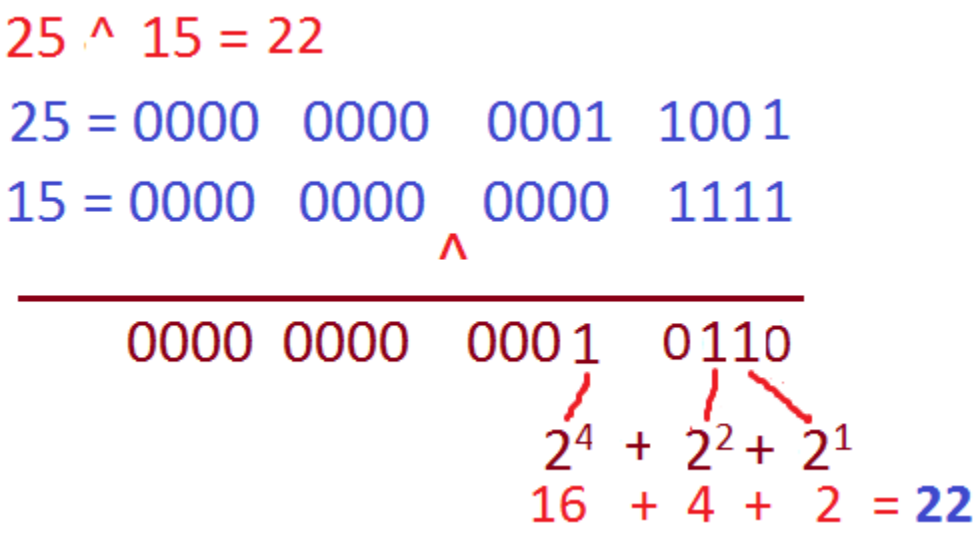
| - Bitwise or: In this both bits are 0's then result bit is 0. Otherwise result bit is 1.

Eg: 25 | 15 = 31



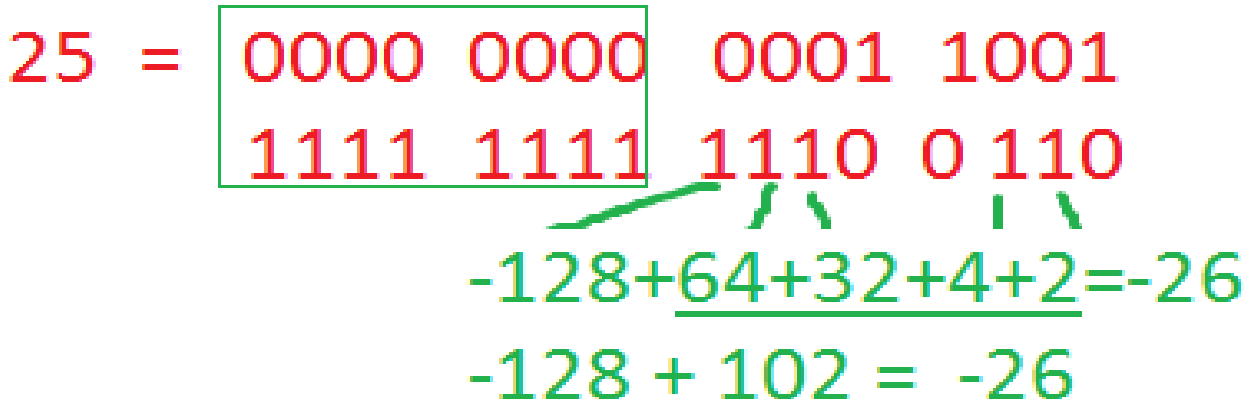
^ - XOR [Exclusive OR]: In this both bits are same then result bit is 0. Otherwise result bit is 1.

Eg: 25 ^ 15 = 22



~ - Compliment operator: In compliment operation the bits are complimented. i.e. 1's become 0's and 0's become 1's. Due to this +Ve no becomes –Ve and –Ve no becomes +Ve.

eg: **~25 ➔ -26**



25 = 0000 0000 0001 1001

~ = 1111 1111 1110 0110

2+4+32+64+128+256+512+1024+2048+4096+8192+16384-32768=-26

~-25 = 0000 0000 0001 1001

1's ~ = 1111 1111 1110 0110

2's ~ = 0000 0000 0000 0001

1111 1111 1110 0111

16+8=24

1 0 0 01

0 1 0 1

1 1 0 10

Note: When starting bit is 1 given no is –Ve.

Eg: ~-25 ➔ +24

~-25 = +24

25 = 0000 0000 0001 1001

1111 1111 1110 0110 <== 1's compliment

+1 <== 2's Compliment

1111 1111 1110 0111

~ 0000 0000 0001 1000

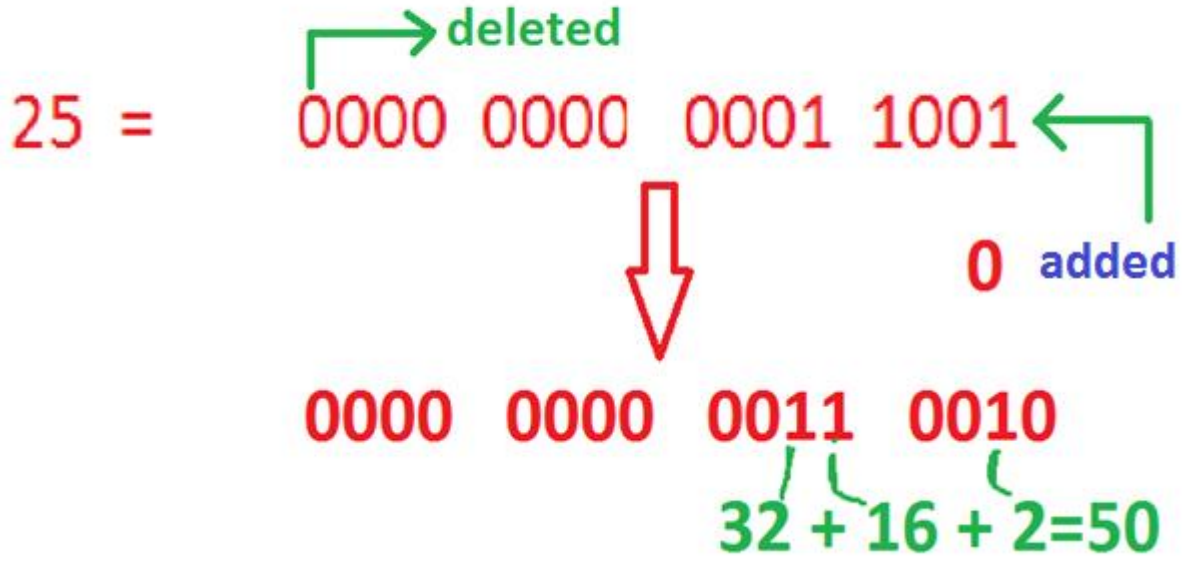
16+8=24

<< - left shift operator:

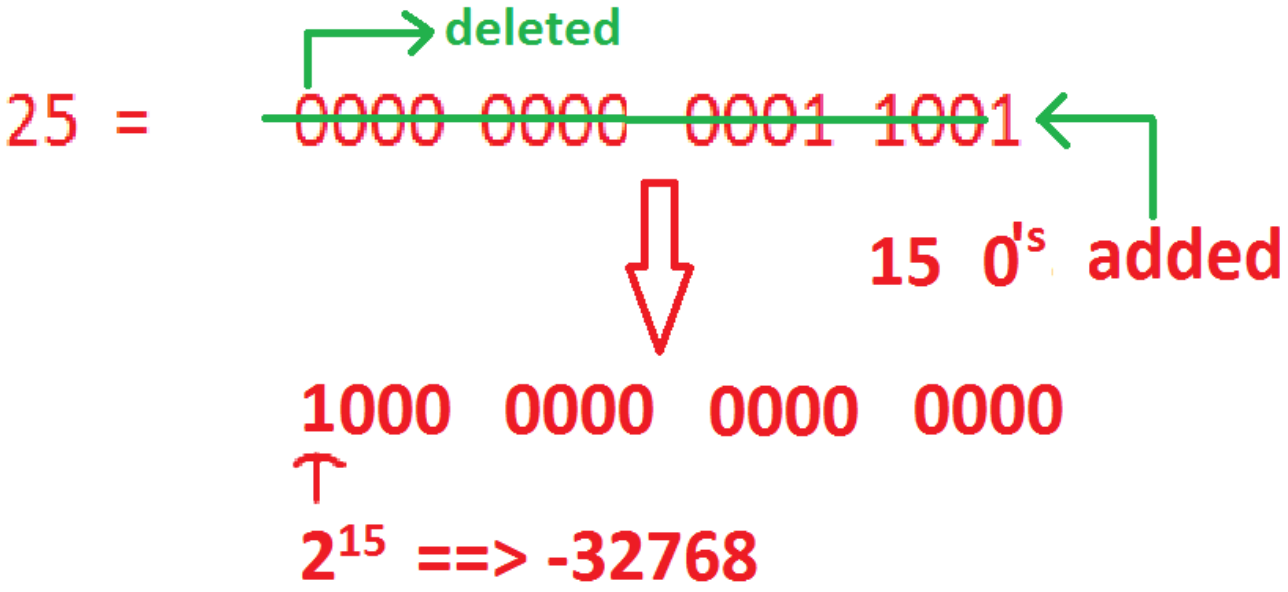
In left shift operation, the specified no of bits are deleted from left side and the same no of zeros added on right side. In left shift operation, most probably the value is multiplied with 2 that no of times.

Eg:25<<1=50, 25<<2=100, 25<<15 =-32768, 25<<16=0

eg: 25<<1=50


deleted
25 = 0000 0000 0001 1001
0 added
0000 0000 0011 0010
32 + 16 + 2=50

eg: 25<<15 = -32768



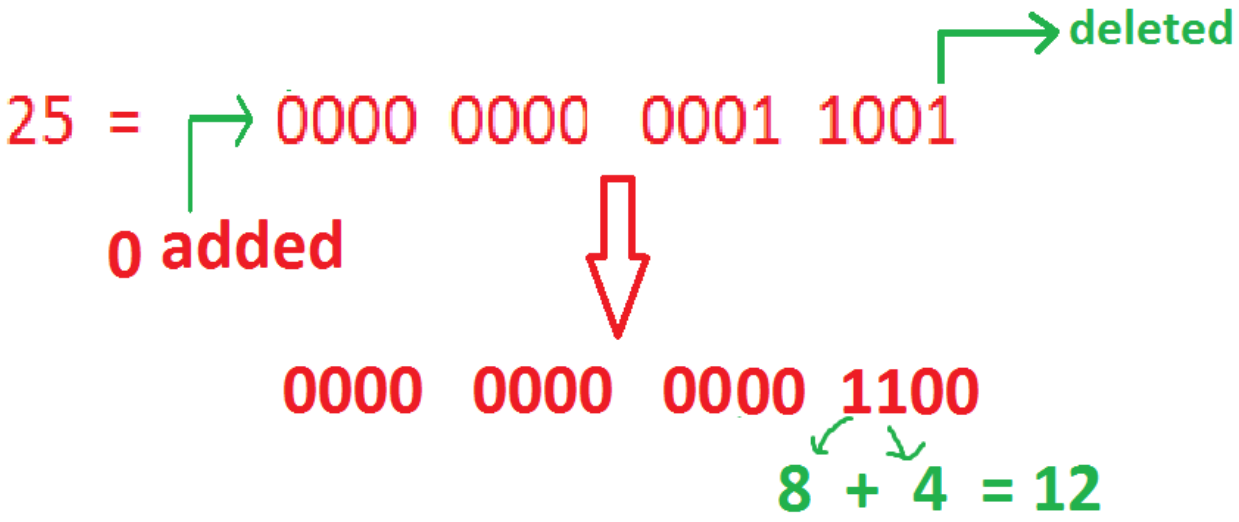
Note: When starting bit 1 no is negative.

>> - Right shift operator:

In right shift operation, the bits are moved to right side i.e. the specified no.of bits are deleted from right side and same no.of zero's are added left side. Due to this always the number is divided with 2 that no of times.

Eg:25>>1=12, 25>>2=6, 25>>3=3, 25>>4=1,25>>5=0

eg: 25 >>1 = 12



eg: 25 >> 5 = 0