(Q) When we want to store a -ve number.

First we are going to convert it into binary string of that number if it was positive and then we are going to find the 2^s complement.

i) 13 → 0 _ _ 1 1 0 1

↑ 31st bit

1s complement of 13 → 1 _ _ 0 0 1 0

Add 1 for making it 2^s complement

2^s complement of 13 →

1 _ _ _ 0 0 1 1

↓ -ve ↓ 2^s complement of 13.

(☆) Largest value that you can store as an integer →

0 1 1 _ _ _ 1 1

↑ last bit ↑ 1st bit

$= 2^{30} + 2^{29} + 2^{28} + \dots 2^0$

$= 2^{31} - 1 = \text{Int\_Max}$

(8)

ii) $\frac{13}{2^2} = 3$ iii) $\frac{13}{2^4} = 0$

→ Why this formula perfectly fits?

13 → 1 1 0 1

1 1 0 1 → $1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

13 >> 1

1 1 0 → $1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

The second binary is obtained by dividing the 1st binary by two.

(*) How does the computer ~~stores~~ differentiates positive and negative number.

The computer stores number in 32 bit or last signed integer. Inside the 31 st bit is used to store the sign.

1 → -ve 0 → +ve

31 st bit (sign).

0 0 0 1 1 0 1 → 13

(7).

0 1 1 0 1 → we are going to remove this 1

0 1 1 0 ≡ 13 >> 1.

↓

6. (integer)

(i) (ii) n = 13 >> 2

0 1 1 0 1

↓

0 1 1 → 3. (integer).

(i) (iii) n = 13 >> 4

0 1 1 0 1

↓

0 0 → 0 (integer)

Formula for right shift operator →
$n >> k = \left(\frac{n}{2^k}\right)$

i) 13 >> 1

$\frac{13}{2^{(1)}} = 6$

(6).

(2) Or Operator →.

1 true means true
all false means false.

(a) n = 13 | 7.

				number represent n
1	1	0	1	13
0	1	1	1	7
(1	1	1	1)$_2$	15

(3) XOR Operator →.

no. of 1s → odd → 1
no. of 1s → even → 0

(a) n = 13 ^ 7.

				number represent n
0	1	1	0	1	13
0	0	1	1	1	7
0	1	0	1	0	10

(4). Shift Operators →.

(a) Right shift operator → (>>)

i). n = 13 >> 1
13 shift by 1.

(5) :.

(b) 2s complement of 7.

(0 0 0) 2 — 1s complement

(0 0 1) — Add 1.

(0 0 1)$_2$ — 2s complement

Operators

(1) And Operator →

- And operator means all true is true.
- One False is false.
- True represents 1 and false represents 0.

(a) n = 13 & 7

13 → (1 1 0 1)$_2$ 7 → (1 1 1)$_2$

1 1 0 1

0 1 1 1

(0 1 0 1)$_2$ ← represent n of 5.

if there is (0,1) or (0,0) it will be considered as 0.

if there is (1,1) then it will be 1

(4). (1's complement)

(a) 1's complement of 13.

$(13) \rightarrow (1101)_2$

↓ flip all the bits

$(0010)_2$

Flip all the bits means flipping all the zeroes to one and all the ones to zeroes.

(b) 1's complement of 7.

$(7) \rightarrow (111)_2$

↓ flip all the bits

$(000)_2$

(2's complement)

Step - 1) Find the 1's complement.
Step 2) Add 1 to it.

(a) 2's complement of 13

$(0010)_2$ 1's complement
(0001) Add 1
$(0011)_2 \rightarrow$ 2's complement

(3).

Zeroes are stored in place of the remaining bits.

And when we are printing the value, computer converts binary format into a number. (decimal).

Computer always converts the number into the binary and then stores it.

For integers 32 bits is used short. we will mostly come across this. atmost

A long int is signed integral type that is atleast 32 bits

A long long is a signed integral type is atleast 64 bits.

The concepts of short bit can be applied to long long bits as they are same.

(2)

3 2 1 0 ←

(c) $(1\ 1\ 0\ 1)_2 \longrightarrow (13)_{10}$

Start from the right and move onto the left.

number X base (index) for each number in the binary equivalent from left to right

$1 \times 2^0 + 0 \times 2^1 + 1 \times 2^2 + 1 \times 2^3$

$1 + 0 + 4 + 8 = (13)_{10}$

(d) $(1\ 1\ 1)_2 \longrightarrow (7)_{10}$

2 1 0

$1 \times 2^0 + 1 \times 2^1 + 1 \times 2^2$

$1 + 2 + 4 = 7$

When we are writing a number on computer such as 13, it doesn't store the number as 13 Instead it stores in the form 1 1 0 1. But, whenever a computer stores a number it stores in 32 bits (int)

(0 0 0 0 ---- 1 1 0 1)

28 bits 4 bits

① Bit Manipulation

✡ Binary Number Conversion (1's & 2's Complement)

ⓐ $(7)_{10} \longrightarrow (111)_2$

7 to the Base 10 (Decimal Number)

Binary Format

2 | 7		1 ↑
2 | 3		1
1		1

Remainder of 7/2
Remainder of 3/2
Remainder as 1 is not divisible by 2.

Consider the remainders as strings and add them in upward direction.

"1" + "1" + "1" = 111

111 is the Binary Format of $(7)_{10}$

ⓑ $(13)_{10} \longrightarrow (1101)_2$

2 | 13		1 ↑
2 | 6		0
2 | 3		1
1		1

"1" + "1" + "0" + "1"
= 1101