## Computer Fundamental

# COMPLEMENT

## Binary Number Divided into two part

i. Signed Binary Number ii. Unsigned Binary Number

## i. Signed Number

- It is represent in +ve/-ve (+/-) integer.
- E.g - …..-3,-2,-1,0,1,2,3,….
- Range of n bit number is = $-(R^n-1)$ to $(R^{n-1}-1)$
- Here R= radix/base , n = bit
- E.g For 3 bit number = $-(2^3-1)$ to $(2^3-1)$

  = -3 to 3

## ii. Unsigned Number

- It is represent only +ve numbers.
- Each bit has 2 possible states = 0 or 1
- So for n bit, total combinations = $2^n$
- Since counting starts form 0, the highest range of number is = $2^n-1$.

✓**Total values = $2^n$**

✓**Total range = 0 to $2^n-1$**

| Bits | Total values ($2^n$) | Range (0 to $2^n-1$) |
|---|---|---|
| 1 bit | 2 | 0 to 1 |
| 2 bits | 4 | 0 to 3 |

| | | |
|---|---|---|
| 3 bits | 8 | 0 to 7 |
| 4 bits | 16 | 0 to 15 |
| 8 bit | 256 | 0 to 255 |
| 32 bit | 4294967296 | 0 to 4294967296 |
| 64 bit | 18,446,744,073,709,551,615 (18.4 quintillion) | |

| Bit size | Type | Range |
|---|---|---|
| 1bit | Unsigned | 0 to 1 |
| | Signed | 0 to 0 |
| 2bit | Unsigned | 0 to 3 |
| | Signed | -2 to 1 |
| 4bit | Unsigned | 0 to 15 |
| | Signed | -8 to 7 |
| 8 bit | Unsigned | 0 to 255 |
| | Signed | -128 to 127 |
| 16 bit | Unsigned | 0 to 65,535 |
| | Signed | -32,768 to 32,767 |

## Q. We can perform subtraction using the binary number system, so why were techniques like sign-magnitude, 1's complement, and 2's complement introduced?

- Because in a computer, there is **no separate circuit for subtraction**.
- A computer only has an **adder circuit**, and the adder performs all operations using addition — even(यहाँ तक की) subtraction.
- Therefore, a method was needed that could **convert subtraction into addition**, so that **one single adder circuit** could perform everything.
- If In computers,

$$A - B = A + (\text{complement of } B)$$

This means: convert **B** into its negative form and then add it.

- The methods that perform this conversion are:

- **Sign-Magnitude**
- **1's Complement**
- **2's Complement**
- My computer find 1's / 2's complement using Logic gates (AND, OR, NOT,NAND,NOR,XOR….).
- My computer solve Subtract ➔use two blocks (i)Adder circuit (Binary adder), (ii) Logic gates

## ❖ Signed Magnitude

- Signed magnitude is a way to represent positive(+ve) and negative(-ve) number in binary by using the leftmost (most significant) bit as the sing bit.
- Here +ve sign use = 0 & -ve sign use = 1
- MSB = Most Significant Bit
- LSB = Least Significant Bit
- If Sign magnitude given that **n** bits then – 1st bit represent sign (-ve/+ve) and remainder (n-1) represent bits magnitude.
  Example
  - ✓ 4 bits (0000) here 1st bit sing (0 then +ve) and remainder (4-1 = 3) bits represent bits magnitude.
  - ✓ 6bits(100000) here 1st bit sing(1 then -ve) and remainder (6-1=5) bits represent bits magnitude.
- Example

| | |
|---|---|
| Two representations of zero<br>+0 = 00000000 (8bit)<br>-0 = 10000000 | Two representation of 25<br>+25 = 011001 (6bit)<br>-25 = 111001 |
| +18 = 00010010 (8bit)<br>-18 = 10010010<br>MSB LSB | +10 = 00001010,<br>-10 = 10001010<br>Left Right |

(+ve number को -ve बनाने के लिए MSB- Most significant bit में सिर्फ 1st वाले bit ( इसे leftmost कहते हैं ) को उल्ट देने से या change कर देने से वह -ve हो जाता हैं | 0 - positive , 1 - negative)

## ➢ Complements

- Complements are used in digital computers to simplify the subtraction operation and for logical manipulation.
- Complement refers to a mathematical operation or concept used in binary arithmetic,digital circuits and computer programming.

## ➢ Use of Complements

- In Subtraction
- Represent of negative number
- In Airthmetic logic unit (ALU)

## ➢ Real-life use of Complements

- For Subtraction in Digital circuits/calculators
- Store negative value in computer memory
- Check Error in CPU
- In networking

## ➢ Two types of Complement

**1. 1's Complement** **2. 2's Complement**

### 1. 1's Complement

- Inverting/Changing all bits of a binary number (0 becomes 1, & 1 becomes 0).
- e.g. – Binary Number : 10110
  1's Complement : 01001

- For '+' → 0 (add in left side) and '-' → 1 (add in left side)
- 1's complement of Positive number is this number/binary not change/reverse bit.

Q1. Find 1's Complement of decimal number +5 ?

→ Binary of +5 is = +101 , (+ के जगह 0 रख दें )

= 0101 Answer

Q2. Find 1's complement of decimal number -37 ?

→ Binary of (+37) = +100101

= 0100101

Now,find 1's complement this number 0100101 ➔ 1011010

Q3. Represent -27 by using 8 bits register in 1's complement

→Binary (+27) = 11011 (in 8 bit → 00011011)

Now, 1's complement of 00011011 → 11100100

## 2. 2's Complement

- Adding 1 (one) to the 1's Complement of a binary number.
- Number of bits is fixed
- e.g -

| | | |
|---|---|---|
| Binary Number | : | 10110 |
| 1's Complement | : | 01001 |
| 01001 + 1 | : | 01010 |
| 2's Complement | : | 01010 (solve) |

## Q. Convert Binary to 2's Complement

| | | |
|---|---|---|
| i) 10100 | : | 01100 |
| ii) 101011 | : | 010101 |
| iii) 110011 | : | 001101 |
| iv) 100110 | : | 011010 |
| v) 0011001100 | : | 1100110100 |

## ❖ Rule of 2's complement addition rule

**Modern Computer use this technic for subtraction any number.**

**Steps**

- Write numbers in binary (take one bit extra in MSB side)
- Convert negative number → **2's complement**
  (1's complement + 1)
- Add both
- If carry comes → **discard (MSB side)**
- Result MSB = 0 then +ve sign and MSB = 1 then find 2's complement again than final value is final answer with -ve sign

# Q1. Add +9 and -5

Step1 → +9 = 01001 ---(i) & -5 =00101
Step2 → 2's complement of -ve number i.e. -5 = 11011-------(ii)
Step 3 → add (i) + (ii) ➔ 01001 + 11011 ➔ 00100 (carry को remove कर दो, i.e. discard)

Q2. (+7)+(-9) {5-bit} ➔ Answer (-2 →00010)
Q3. (-6) + (-5) {5 -bit} ➔ Answer (-11 → 01011)
Q4. (+12) + )-7) {6 bit} ➔ Answer (+5 → 000101)
Q5. (-8) + (+3) {5 bit} ➔ Answer (-5 → 00101)
Q6. (+15) + (-15) {6 bit} ➔ Answer (0 )
Q7. (-10) + (+6) {5-bit} ➔ Answer (-4 → 00100)

## ❖ Rule of Multiplication

1. Decimal to Binary
2. Start with Multiplier (नीचे वाला) {bit 1 then write same number & bit 0 then write 0. Again every line left shift (*2) Example – bit 1 → number , bit 0 → 000
3. Add all rows
4. Result

# Q1. 5 * 3

Step 1 ➔ 5 in binary = 101 , 3 in binary = 011
Step ➔ start with multiplier i.e. 011 (Read Right to left)
If First bit = 1 → 101 (same number)
If Second bit 1 (one left shift) → 101(same number)
If Third bit 0 (2 left shift) → 000
Step ➔ Add all 101
101 →(one left shift)
000 → (two left shift)
==========
1111
Step ➔ Answer 15 (1111)

========== HIRA KUMAR ==========