Spondon Ganguli
An Educator, Artist & Author

BINARY NUMBER SYSTEM

- ❖ Binary Arithmetic
  - Addition
  - Subtraction
  - Multiplication
  - Division
- ❖ Negative number presentation
- ❖ Floating point presentation

BINARY ARITHMETIC

Addition	**0+0 = 0**	**Sum = 0**	**Carry = 0**
	0+1 = 1	**Sum = 1**	**Carry = 0**
	1+0 = 1	**Sum = 1**	**Carry = 0**
	1+1 = 10	**Sum = 0**	**Carry = 1**
	1+1+1=11	**Sum = 1**	**Carry = 1**
Subtraction	**0-0 = 0**	**Diff = 0**	**Borrow = 0**
	10-1 = 1	**Diff = 1**	**Borrow = 1**
	1-0 = 1	**Diff = 1**	**Borrow = 0**
	1-1 = 0	**Diff = 0**	**Borrow = 0**
Multiplication	**0*0=0*1=1*0 = 0**		
	1*1 = 1		
Division	**0/0 = 0**		
	0/1 = 0		
	1/0 = NAN		
	1/1 = 1		

Spondon Ganguli
An Educator, Artist & Author

Some examples of Binary Arithmetic:

Here all the numbers are in binary form.

Addition	**1010+1111 = 11001**	**1010.011+110.110 = 10001.001**
	(Carry)1110 **1010** **+1111** **----------** **11001**	***(Carry)11 1 100*** **1010.011** **+0110.110** **------------------** **10001.001**
Subtraction	**11001 – 1111 = 1010**	**10001.001 – 1010.011 = 110.110**
	(Borrow) 0 10 1 10 0 ~~10~~ ~~1~~ ~~1~~ 0 0 1 (-) 1 1 1 1 0 1 0 1 0	**(Borrow)** 1 10 0 1 ~~10~~ ~~10~~ 1 10 0 ~~10~~ ~~1~~ 0 0 0 ~~1~~.0 0 1 (-) 1 0 1 0.0 1 1 0 0 1 1 0.1 1 0
Multiplication	**101x101 = 11001**	**101.1 x 11.01 = 10001.1110**
	101 **(x) 101** **101** **000x** **(+) 101xx** **11001**	**101.10** **(x) 011.01** **10110** **00000x** **10110xx** **(+) 10110xxx** **10001.1110**
Division	**11001 ÷ 101 = 101**	
	101 | 11001 | 101 **(-) 101** **101** **(-) 101** **xxx**	

Spondon Ganguli (Follow me on Youtube for lectures on Java programming: https://www.youtube.com/channel/UCatZ4RzjcwFVOBQKSwmkOZw)

Spondon Ganguli
An Educator, Artist & Author

TOPIC: Computer Number System

COMPLEMENT OF A NUMBER

Complement of a number is another number adding that to the original number we get the highest digits in each position.

In **decimal number system**, we have 9'c complement and 10's complement.
So, 9's complement of $(123.45)_{10}$ will be $(876.54)_{10}$.
Adding $(123.45)_{10}$ with $(876.54)_{10}$ will give the result $(999.99)_{10}$
10's complement = 9's complement + 1 (with the Least Significant Digit)
So, 10's complement of $(123.45)_{10}$ will be $(876.55)_{10}$

Binary number system has two complements – 1's complement and 2's complement

1. **1's Complement** – converting every 1 to 0 and 0 to 1
   For example, 1's complement of $(1001.1011)_2 = (0110.0100)_2$
2. **2's Complement –** 1's complement + 1 (at the LSB position)
   For example, 2's complement of $(1001.1011)_2 = (0110.0101)_2$

In **octal number system**, we have 7'c complement and 8's complement.
So, 7's complement of $(123.45)_8$ will be $(654.32)_8$.
Adding $(123.45)_{10}$ with $(654.32)_{10}$ will give the result $(777.77)_8$
8's complement = 7's complement + 1 (with the Least Significant Digit)
So, 8's complement of $(123.45)_{10}$ will be $(654.33)_8$.

In **hexadecimal number system**, we have15'c complement and 16's complement.
So, 15's complement of $(A2B.45)_{16}$ will be $(5D4.BA)_{16}$
Adding $(A2B.45)_{16}$ with $(5D4.BA)_{16}$ will give the result $(FFF.FF)_{16}$
16's complement = 15's complement + 1 (with the Least Significant Digit)
So, 16's complement of $(A2B.45)_{10}$ will be $(5D4.BB)_{10}$

Least Significant Digit – It is the digit which has lowest positional value in the number.
Most Significant Digit – It is the digit which has highest positional value in the number.
For example, if the number is $(123.45)_{10}$ then the MSD is 1 (with 10^3 positional value) and LSD is 5 (with 10^{-2} positional value).

BINARY SUBTRACTION USING 1'S COMPLEMENT & 2'S COMPLEMENT METHODS:

Steps for Binary subtraction using 1's complement:

Let us take two binary numbers, A= $(1010.11)_2$ and B= $(110.011)_2$

Step 1: Balance both the numbers A and B.
A = 1010.110
B = 0110.011

Step 2: Get one's complement of B
1001.100

Step 3: Add A with 1's complement of B

```
  1010.110
  1001.100
1 0100.010
```

Step 4: If we get a carry in the sum, then add it with the LSB of the sum.

```
0100.010
       1
0100.011
```

Thus $(0100.011)_2$ is the final answer.

Step 4 (b): If there is no carry then, get one's complement of the sum and that will be the answer with a minus (-) sign.

For example, A = 110 and B = 1001
Step 1: 0110 – 1001
Step 2: 1's complement of 1001 = 0110
Step 3: 0110 + 0110 = 1100
Step 4: Since there is no carry, then we have to take the 1's complement of the sum

$\therefore$ Answer will be $-(0011)_2$

Steps for Binary subtraction using 2's complement:

Let us take two binary numbers, A= $(1010.11)_2$ and B= $(110.011)_2$

Step 1: Balance both the numbers A and B.

A = 1010.110

B = 0110.011

Step 2: Get two's complement of B (001.100+0.001)

1001.101

Step 3: Add A with 1's complement of B

```
  1010.110
  1001.101
----------
1 0100.011
```

Step 4: If we get a carry in the sum, then ignore the carry and the rest is the answer.

0100.011

Thus $(0100.011)_2$ is the final answer.

Step 4 (b): If there is no carry then, get two's complement of the sum and that will be the answer with a minus (-) sign.

For example, A = 110 and B = 1001

Step 1: 0110 – 1001

Step 2: 2's complement of 1001 = 0111

Step 3: 0110 + 0111 = 1101

Step 4: Since there is no carry, then we have to take the 2's complement of the sum

$\therefore$ Answer will be $-(0011)_2$

An Educator, Artist & Author

NEGATIVE NUMBER REPRESENTATIN IN BINARY NUMBER SYSTEM

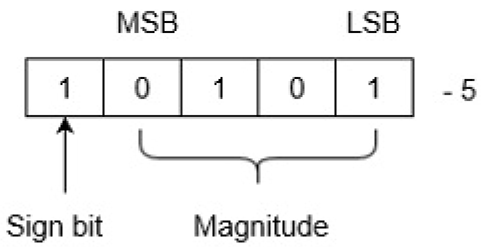
Negative numbers can be distinguishable with the help of extra bit or flag called sign bit or sign flag in Binary number representation system for signed numbers. It is not possible to add minus or plus symbol in front of a binary number because a binary number can have only two symbol either 0 or 1 for each position or bit. That's why we use this extra bit called sign bit or sign flag. The value of sign bit is 1 for negative binary numbers and 0 for positive numbers.

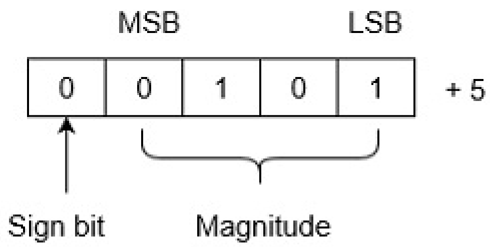
When an integer binary number is positive, the sign is represented by 0 and the magnitude by a positive binary number. When the number is negative, the sign is represented by 1 but the rest of the number may be represented in one of three possible ways: Sign-Magnitude method, 1's Complement method, and 2's complement method. These are explained as following below.

1. Signed Magnitude Method:

In this method, number is divided into two parts: Sign bit and Magnitude. If the number is positive then sign bit will be 0 and if number is negative then sign bit will be 1. Magnitude is represented with the binary form of the number to be represented.

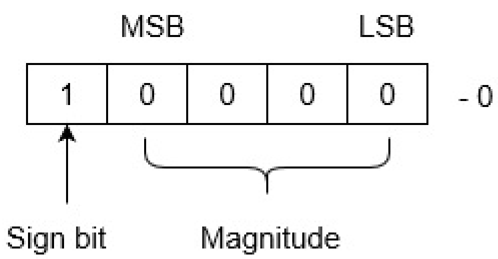
Example: Let we are using 5 bits register. The representation of -5 to +5 will be as follows:

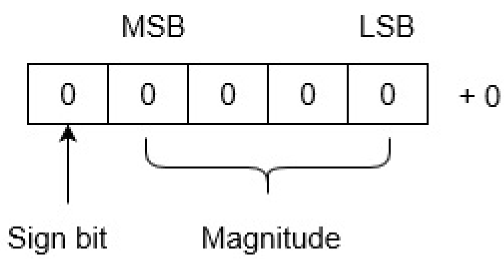




Range of Numbers: For k-bits register, MSB will be sign bit and (k-1) bits will be magnitude. Positive largest number that can be stored is $(2^{(k-1)}-1)$ and negative lowest number that can be stored is $-(2^{(k-1)}-1)$.

Note that drawback of this system is that 0 has two different representation one is -0 (e.g., 1 0000 in five bit register) and second is +0 (e.g., 0 0000 in five bit register).

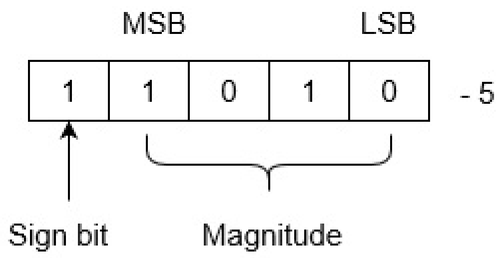


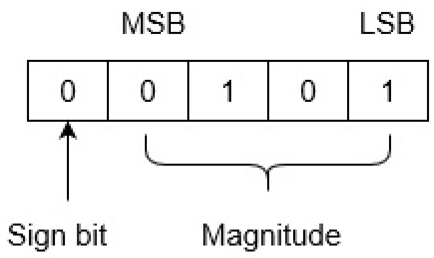


2. 1's Complement Method:

Positive numbers are represented in the same way as they are represented in sign magnitude method. If the number is negative then it is represented using 1's complement. First represent the number with positive sign and then take 1's complement of that number.

Example: Let we are using 5 bits register. The representation of -5 and +5 will be as follows:





+5 is represented as it is represented in sign magnitude method. -5 is represented using the following steps:

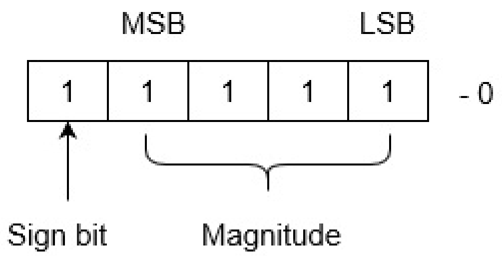
(i) +5 = 0 0101

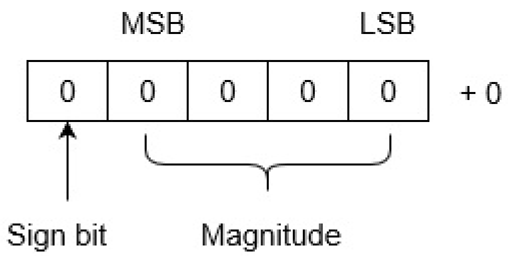
(ii) Take 1's complement of 0 0101 and that is 1 1010. MSB is 1 which indicates that number is negative.

MSB is always 1 in case of negative numbers.

Range of Numbers: For k-bits register, positive largest number that can be stored is ($2^{(k-1)}$-1) and negative lowest number that can be stored is -($2^{(k-1)}$-1).

Note that drawback of this system is that 0 has two different representation one is -0 (e.g., 1 1111 in five bit register) and second is +0 (e.g., 0 0000 in five bit register).

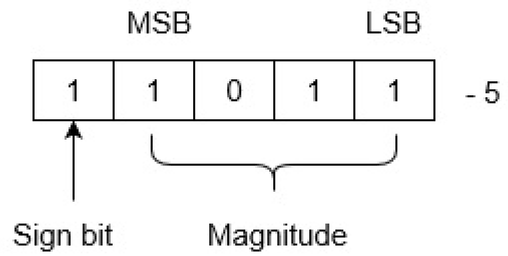


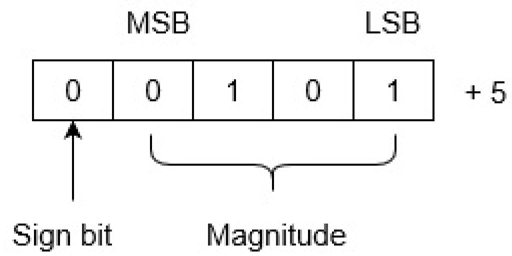


3. 2's Complement Method

Positive numbers are represented in the same way as they are represented in sign magnitude method. If the number is negative then it is represented using 2's complement. First represent the number with positive sign and then take 2's complement of that number.

Example: Let we are using 5 bits registers. The representation of -5 and +5 will be as follows:





+5 is represented as it is represented in sign magnitude method. -5 is represented using the following steps:

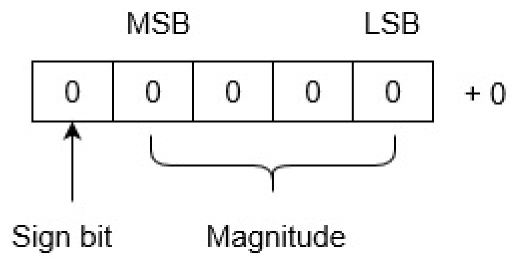
(i) +5 = 0 0101

(ii) Take 2's complement of 0 0101 and that is 1 1011. MSB is 1 which indicates that number is negative.

MSB is always 1 in case of negative numbers.

Range of Numbers: For k bits register, positive largest number that can be stored is $(2^{(k-1)}-1)$ and negative lowest number that can be stored is $-(2^{(k-1)})$.

The advantage of this system is that 0 has only one representation for -0 and +0. Zero (0) is considered as always positive (sign bit is 0) in 2's complement representation. Therefore, it is unique or unambiguous representation.



These are representation method for signed binary numbers.

FLOATING POINT BINARY NUMBERS

Floating point Binary numbers are represented in **Mantissa-Exponent** form. In mantissa-exponent form of data, a number has two part : **mantissa** and **exponent**. The mantissa is the number with decimal point and the exponent is the value of power to generate the actual number.
For example,
In decimal system: $(42.576)_{10} = 0.42576 \times 10^2$ i.e., 0.42576E2
In binary system: $(100101.1)_2 = 0.1001011 \times 2^6$ i.e., $0.1001011E^{110}$

Now let us see how to convert a decimal number to binary in n-bit Floating point system?
Express $(17.125)_{10}$ in a binary floating point notation on 16-bit system:

Step 1: Convert $(17.125)_{10}$ to binary
$(10001.001)_2$
Step 2: Relocate the point and write the mantissa (in the form $0.M \times 2^E$)
0.10001001×2^5
Step 3: Since exponent is positive, represent that in binary form
$E = 101_2$
Step 4: Segregate the number in Sign, Exponent and Mantissa
Sign (S) = 0 (for positive)
Exponent (E) = 101 Mantissa (M) = 10001001

0	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**0**
1	3	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Sign	Exponent				Mantissa									

Answer: $(17.125)_{10}$ = $(001011000100100)_2$ in 16 bit Binary floating point notation.

Express $(-0.125)_{10}$ in a binary floating point notation on 16-bit system:
Step 1: Convert $(-0.125)_{10}$ to binary
$(-0.001)_2$
Step 2: Relocate the point and write the mantissa
0.1×2^{-2}
Step 3: Since exponent is negative, get it's two's complement in 4-bit system
$(2)_{10} = (0010)_2$
Two's complement = 1101+1 = 1110_2
Step 4: Segregate the number in Sign, Exponent and Mantissa
Sign (S) = 1 (for negative)
Exponent (E) = 1110 Mantissa (M) = 1

1	**1**	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
1	3	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Sign	Exponent				Mantissa									

Answer: $(-0.125)_{10}$ = $(11101000000000)_2$ in 16 bit Binary floating point notation.

Spondon Ganguli (Follow me on Youtube for lectures on Java programming:
https://www.youtube.com/channel/UCatZ4RzjcwFVOBQKSwmkOZw)