Complements are used in the digital computers in order to simplify the subtraction operation and for the logical manipulations. For each radix-r system (radix r represents base of number system) there are two types of complements.

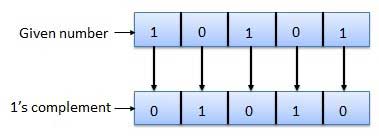
|  |  |  |
| --- | --- | --- |
| **S.N.** | **Complement** | **Description** |
| 1 | Radix Complement | The radix complement is referred to as the r's complement |
| 2 | Diminished Radix Complement | The diminished radix complement is referred to as the (r-1)'s complement |

Binary system complements

As the binary system has base r = 2. So the two types of complements for the binary system are 2's complement and 1's complement.

1's complement

The 1's complement of a number is found by changing all 1's to 0's and all 0's to 1's. This is called as taking complement or 1's complement. Example of 1's Complement is as follows.

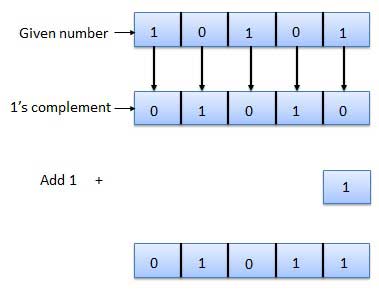


2's complement

The 2's complement of binary number is obtained by adding 1 to the Least Significant Bit (LSB) of 1's complement of the number.

2's complement = 1's complement + 1

Example of 2's Complement is as follows.



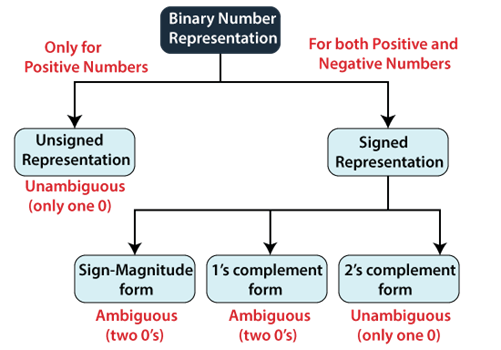
# **Signed and Unsigned Binary Numbers**

The integer variables are represented in a signed and unsigned manner. The positive and negative values are differentiated by using the sign flag in signed numbers. The unsigned numbers do not use any flag for the sign, i.e., only positive numbers can be stored by the unsigned numbers.

It is very easy to represent positive and negative numbers in our day to day life. We represent the positive numbers without adding any sign before them and the negative number with - (minus) sign before them. But in the digital system, it is not possible to use negative sign before them because the data is in binary form in digital computers. For representing the sign in binary numbers, we require a special notation.

## Binary Numbers Representation

Our computer can understand only (0, 1) language. The binary numbers are represented in both ways, i.e., signed and unsigned. The positive numbers are represented in both ways- signed and unsigned, but the negative numbers can only be described in a signed way. The difference between unsigned and signed numbers is that unsigned numbers do not use any sign bit for positive and negative numbers identification, but the signed number used.



### Unsigned Numbers

As we already know, the unsigned numbers don't have any sign for representing negative numbers. So the unsigned numbers are always positive. By default, the decimal number representation is positive. We always assume a positive sign in front of each decimal digit.

38M

869

OOPs Concepts in Java

There is no sign bit in unsigned binary numbers so it can only represent its magnitude. In zero and one, zero is an unsigned binary number. There is only one zero (0) in this representation, which is always positive. Because of one unique binary equivalent form of a number in unsigned number representation, it is known as unambiguous representation technique. The range of the unsigned binary numbers starts from 0 to (2n-1).

**Example:** Represent the decimal number 102 in unsigned binary numbers.

We will change this decimal number into binary, which has the only magnitude of the given name.

|  |  |  |  |
| --- | --- | --- | --- |
| **Decimal** | **Operation** | **Result** | **Remainder** |
| 102 | 102/2 | 51 | 0 |
| 51 | 51/2 | 25 | 1 |
| 25 | 25/2 | 12 | 1 |
| 12 | 12/2 | 6 | 0 |
| 6 | 6/2 | 3 | 0 |
| 3 | 3/2 | 1 | 1 |
| 1 | 1/2 | 0 | 1 |

So the binary number of (102)10 is (1100110)2, a 7-bit magnitude of the decimal number 102.

### Signed Numbers

The signed numbers have a sign bit so that it can differentiate positive and negative integer numbers. The signed binary number technique has both the sign bit and the magnitude of the number. For representing the negative decimal number, the corresponding symbol in front of the binary number will be added.

The signed numbers are represented in three ways. The signed bit makes two possible representations of zero (positive (0) and negative (1)), which is an ambiguous representation. The third representation is 2's complement representation in which no double representation of zero is possible, which makes it unambiguous representation. There are the following types of representation of signed binary numbers:

1. **Sign-Magnitude form**

In this form, a binary number has a bit for a sign symbol. If this bit is set to 1, the number will be negative else the number will be positive if it is set to 0. Apart from this sign-bit, the n-1 bits represent the magnitude of the number.

1. **1's Complement**

By inverting each bit of a number, we can obtain the 1's complement of a number. The negative numbers can be represented in the form of 1's complement. In this form, the binary number also has an extra bit for sign representation as a sign-magnitude form.

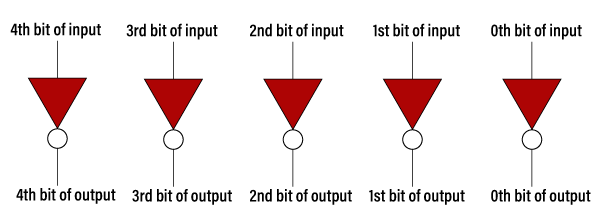
1. **2's Complement**

By inverting each bit of a number and adding plus 1 to its least significant bit, we can obtain the 2's complement of a number. The negative numbers can also be represented in the form of 2's complement. In this form, the binary number also has an extra bit for sign representation as a sign-magnitude form.

# **1's complement**

In number representation techniques, the binary number system is the most used representation technique in digital electronics. The complement is used for representing the negative decimal number in binary form. Different types of complement are possible of the binary number, but 1's and 2's complements are mostly used for binary numbers. We can find the 1's complement of the binary number by simply inverting the given number. For example, 1's complement of binary number 1011001 is 0100110. We can find the 2's complement of the binary number by changing each bit(0 to 1 and 1 to 0) and adding 1 to the least significant bit. For example, 2's complement of binary number 1011001 is (0100110)+1=0100111.

For finding 1's complement of the binary number, we can implement the logic circuit also by using NOT gate. We use NOT gate for each bit of the binary number. So, if we want to implement the logic circuit for 5-bit 1's complement, five NOT gates will be used.



**Example 1: 11010.1101**

For finding 1's complement of the given number, change all 0's to 1 and all 1's to 0. So the 1's complement of the number 11010.1101 comes out **00101.0010**.

32.3M

644

HTML Tutorial

**Next**

**Stay**

**Example 2: 100110.1001**

For finding 1's complement of the given number, change all 0's to 1 and all 1's to 0. So, the 1's complement of the number 100110.1001 comes out **011001.0110**.

### 1's Complement Table

|  |  |
| --- | --- |
| **Binary Number** | **1's Complement** |
| 0000 | 1111 |
| 0001 | 1110 |
| 0010 | 1101 |
| 0011 | 1100 |
| 0100 | 1011 |
| 0101 | 1010 |
| 0110 | 1001 |
| 0111 | 1000 |
| 1000 | 0111 |
| 1001 | 0110 |
| 1010 | 0101 |
| 1011 | 0100 |
| 1100 | 0011 |
| 1101 | 0010 |
| 1110 | 0001 |
| 1111 | 0000 |

## Use of 1's complement

1's complement plays an important role in representing the signed binary numbers. The main use of 1's complement is to represent a signed binary number. Apart from this, it is also used to perform various arithmetic operations such as addition and subtraction.

In signed binary number representation, we can represent both positive and negative numbers. For representing the positive numbers, there is nothing to do. But for representing negative numbers, we have to use 1's complement technique. For representing the negative number, we first have to represent it with a positive sign, and then we find the 1's complement of it.

Let's take an example of a positive and negative number and see how these numbers are represented.

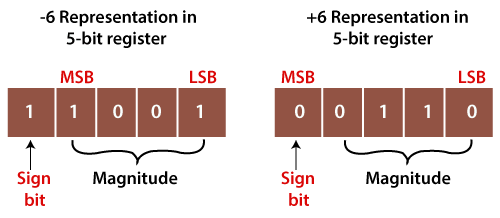
**Example 1: +6 and -6**

The number +6 is represented as same as the binary number. For representing both numbers, we will take the 5-bit register.

So the +6 is represented in the 5-bit register as 0 0110.

The -6 is represented in the 5-bit register in the following way:

1. +6=0 0110
2. Find the 1's complement of the number 0 0110, i.e., 1 1001. Here, MSB denotes that a number is a negative number.



Here, MSB refers to Most Significant Bit, and LSB denotes the Least Significant Bit.

**Example 2: +120 and -120**

The number +120 is represented as same as the binary number. For representing both numbers, take the 8-bit register.

So the +120 is represented in the 8-bit register as 0 1111000.

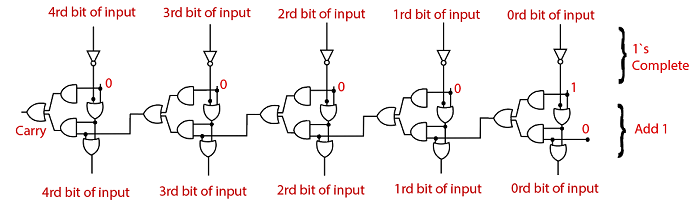
The -120 is represented in the 8-bit register in the following way:

1. +120=0 1111000
2. Now, find the 1's complement of the number 0 1111000, i.e., 1 0000111. Here, the MSB denotes the number is the negative number.

# **2's complement**

Just like 1's complement, 2's complement is also used to represent the signed binary numbers. For finding 2's complement of the binary number, we will first find the 1's complement of the binary number and then add 1 to the least significant bit of it.

For example, if we want to calculate the 2's complement of the number 1011001, then firstly, we find the 1's complement of the number that is 0100110 and add 1 to the LSB. So, by adding 1 to the LSB, the number will be (0100110)+1=0100111. We can also create the logic circuit using OR, AND, and NOT gates. The logic circuit for finding 2's complement of the 5-bit binary number is as follows:



**Example 1: 110100**

For finding 2's complement of the given number, change all 0's to 1 and all 1's to 0. So the 1's complement of the number 110100 is 001011. Now add 1 to the LSB of this number, i.e., (001011)+1=001100.

32.6M

626

Java Try Catch

**Example 2: 100110**

For finding 1's complement of the given number, change all 0's to 1 and all 1's to 0. So, the 1's complement of the number 100110 is 011001. Now add one the LSB of this number, i.e., (011001)+1=011010.

### 2's Complement Table

|  |  |  |
| --- | --- | --- |
| **Binary Number** | **1's Complement** | **2's complement** |
| 0000 | 1111 | 0000 |
| 0001 | 1110 | 1111 |
| 0010 | 1101 | 1110 |
| 0011 | 1100 | 1101 |
| 0100 | 1011 | 1100 |
| 0101 | 1010 | 1011 |
| 0110 | 1001 | 1010 |
| 0111 | 1000 | 1001 |
| 1000 | 0111 | 1000 |
| 1001 | 0110 | 0111 |
| 1010 | 0101 | 0110 |
| 1011 | 0100 | 0101 |
| 1100 | 0011 | 0100 |
| 1101 | 0010 | 0011 |
| 1110 | 0001 | 0010 |
| 1111 | 0000 | 0001 |

### Use of 2's complement

2's complement is used for representing signed numbers and performing arithmetic operations such as subtraction, addition, etc. The positive number is simply represented as a magnitude form. So there is nothing to do for representing positive numbers. But if we represent the negative number, then we have to choose either 1's complement or 2's complement technique. The 1's complement is an ambiguous technique, and 2's complement is an unambiguous technique. Let's see an example to understand how we can calculate the 2's complement in signed binary number representation.

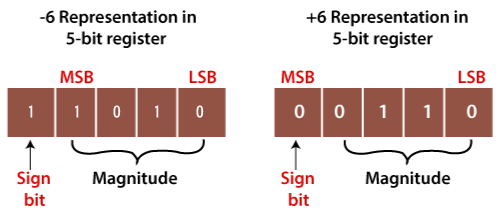
**Example 1: +6 and -6**

The number +6 is represented as same as the binary number. For representing both numbers, take the 5-bit register.

So the +6 is represented in the 5-bit register as 0 0110.

The -6 is represented in the 5-bit register in the following way:

1. +6=0 0110
2. Now, find the 1's complement of the number 0 0110, i.e. 1 1001.
3. Now, add 1 to its LSB. When we add 1 to the LSB of 11001, the newly generated number comes out 11010. Here, the sign bit is one which means the number is the negative number.



**Example 2: +120 and -120**

The number +120 is represented as same as the binary number. For representing both numbers, take the 8-bit register.

So the +120 is represented in the 8-bit register as 0 1111000.

The -120 is represented in the 8-bit register in the following way:

1. +120=0 1111000
2. Now, find the 1's complement of the number 0 1111000, i.e. 1 0000111. Here, the MSB denotes the number is the negative number.
3. Now, add 1 to its LSB. When we add 1 to the LSB of 1 0000111, the newly generated number comes out 1 0001000. Here, the sign bit is one, which means the number is the negative number.