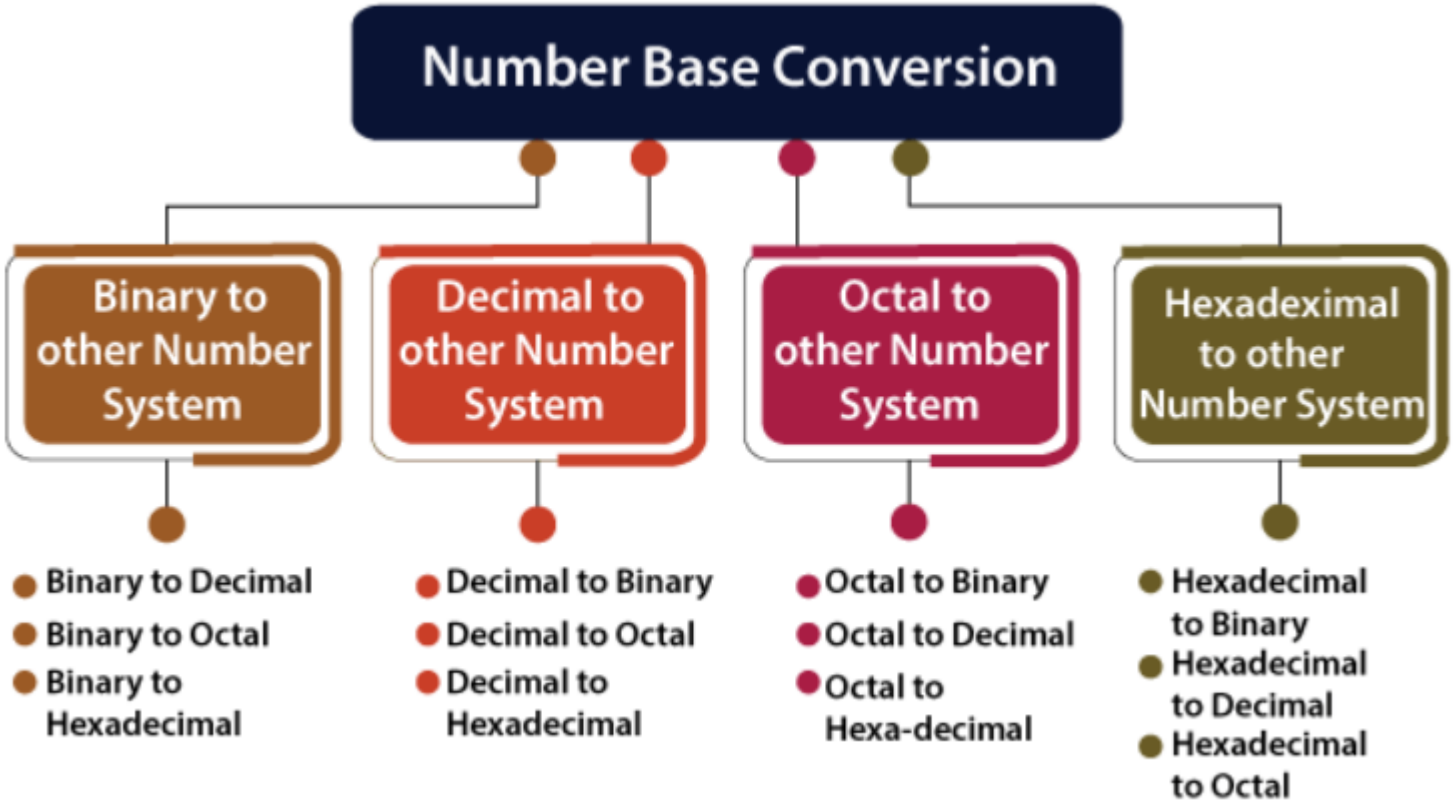
Module 1: NUMBER SYSTEMS

Binary, Octal, Decimal, Hexadecimal. Number base conversions, 1’s, 2’s complements, signed Binary numbers. Binary Arithmetic, Binary codes: Weighted BCD, Gray code, Excess 3 code, ASCII.

Number Base Conversion

There are the following conversions possible in Number System

1. Binary to other Number Systems.
2. Decimal to other Number Systems.
3. Octal to other Number Systems.
4. Hexadecimal to other Number Systems.



Binary to Decimal Conversion

The process of converting binary to decimal is quite simple. The process starts from multiplying the bits of binary number with its corresponding positional weights. And lastly, we add all those products.

Let's take an example to understand how the conversion is done from binary to decimal.

Example 1: $(10110.001)_2$

We multiplied each bit of $(10110.001)_2$ with its respective positional weight, and last we add the products of all the bits with its weight.

$$\mathbf{(10110.001)_2}=(1\times2^4)+(0\times2^3)+(1\times2^2)+(1\times2^1)+(0\times2^0)+$$
$$(0\times2^{-1})+(0\times2^{-2})+(1\times2^{-3})$$
$$\mathbf{(10110.001)_2}=(1\times16)+(0\times8)+(1\times4)+(1\times2)+(0\times1)+$$
$$(0\times½)+(0\times¼)+(1\times⅛)$$
$$\mathbf{(10110.001)_2}=16+0+4+2+0+0+0+0.125$$
$$\mathbf{(10110.001)_2}=\mathbf{(22.125)_{10}}$$

Binary to Octal Conversion

The base numbers of binary and octal are 2 and 8, respectively. In a binary number, the pair of three bits is equal to one octal digit. There are only two steps to convert a binary number into an octal number which are as follows:

1. In the first step, we have to make the pairs of three bits on both sides of the binary point. If there will be one or two bits left in a pair of three bits pair, we add the required number of zeros on extreme sides.
2. In the second step, we write the octal digits corresponding to each pair.

Example 1: $(111110101011.0011)_2$

1. Firstly, we make pairs of three bits on both sides of the binary point.

111 110 101 011.001 1

On the right side of the binary point, the last pair has only one bit. To make it a complete pair of three bits, we added two zeros on the extreme side.

111 110 101 011.001 100

2. Then, we wrote the octal digits, which correspond to each pair.

$(111110101011.0011)_2=(7653.14)_8$

Binary to Hexadecimal Conversion

The base numbers of binary and hexadecimal are 2 and 16, respectively. In a binary number, the pair of four bits is equal to one hexadecimal digit. There are also only two steps to convert a binary number into a hexadecimal number which are as follows:

1. In the first step, we have to make the pairs of four bits on both sides of the binary point. If there will be one, two, or three bits left in a pair of four bits pair, we add the required number of zeros on extreme sides.
2. In the second step, we write the hexadecimal digits corresponding to each pair.

Example 1: $(10110101011.0011)_2$

1. Firstly, we make pairs of four bits on both sides of the binary point.

111 1010 1011.0011

On the left side of the binary point, the first pair has three bits. To make it a complete pair of four bits, add one zero on the extreme side.

0111 1010 1011.0011

2. Then, we write the hexadecimal digits, which correspond to each pair.

$(011110101011.0011)_2=(7AB.3)_{16}$

Decimal to other Number System

The decimal number can be an integer or floating-point integer. When the decimal number is a floating-point integer, then we convert both part (integer and fractional) of the decimal number in the isolated form(individually). There are the following steps that are used to convert the decimal number into a similar number of any base **'r'**.

1. In the first step, we perform the division operation on integer and successive part with base **'r'**. We will list down all the remainders till the quotient is zero. Then we find out the remainders in reverse order for getting the integer part of the equivalent number of base **'r'**. In this, the least and most significant digits are denoted by the first and the last remainders.
2. In the next step, the multiplication operation is done with base **'r'** of the fractional and successive fraction. The carries are noted until the result is zero or when the required number of the equivalent digit is obtained. For getting the fractional part of the equivalent number of base **'r'**, the normal sequence of carrying is considered.

Decimal to Binary Conversion

For converting decimal to binary, there are two steps required to perform, which are as follows:

1. In the first step, we perform the division operation on the integer and the successive quotient with the base of binary(2).
2. Next, we perform the multiplication on the integer and the successive quotient with the base of binary(2).

Example 1: $(152.25)_{10}$

Step 1:

Divide the number 152 and its successive quotients with base 2.

Operation	Quotient	Remainder
152/2	76	0 (LSB)
76/2	38	0
38/2	19	0
19/2	9	1
9/2	4	1
4/2	2	0
2/2	1	0
1/2	0	1(MSB)

$(152)_{10}=(10011000)_2$

Step 2:

Now, perform the multiplication of 0.27 and successive fraction with base 2.

Operation	Result	carry
0.25×2	0.50	0
0.50×2	0	1

$(0.25)_{10}=(.01)_2$

Decimal to Octal Conversion

For converting decimal to octal, there are two steps required to perform, which are as follows:

1. In the first step, we perform the division operation on the integer and the successive quotient with the base of octal(8).
2. Next, we perform the multiplication on the integer and the successive quotient with the base of octal(8).

Example 1: $(152.25)_{10}$

Step 1:

Divide the number 152 and its successive quotients with base 8.

Operation	Quotient	Remainder
152/8	19	0
19/8	2	3
2/8	0	2

$(152)_{10}=(230)_8$

Step 2:

Now perform the multiplication of 0.25 and successive fraction with base 8.

Operation	Resu t	carry
0.25×8	**0**	**2**

$(0.25)_{10}=(2)_8$

So, the octal number of the decimal number 152.25 is **230.2**

Decimal to hexadecimal conversion

For converting decimal to hexadecimal, there are two steps required to perform, which are as follows:

1. In the first step, we perform the division operation on the integer and the successive quotient with the base of hexadecimal (16).
2. Next, we perform the multiplication on the integer and the successive quotient with the base of hexadecimal (16).

Example 1: $(152.25)_{10}$

Step 1:

Divide the number 152 and its successive quotients with base 8.

Operation	Quotient	Remainder
152/16	9	8
9/16	0	9

Step 2:

Now perform the multiplication of 0.25 and successive fraction with base 16.

Operation	Result	carry
0.25×16	0	4

$(0.25)_{10}=(4)_{16}$

$(152)_{10}=(98)_{16}$

So, the hexadecimal number of the decimal number 152.25 is **230.4.**

Octal to other Number System

Like binary and decimal, the octal number can also be converted into other number systems. The process of converting octal to decimal differs from the remaining one.

Octal to Decimal Conversion

The process of converting octal to decimal is the same as binary to decimal. The process starts from multiplying the digits of octal numbers with its corresponding positional weights. And lastly, we add all those products.

Example 1: $(152.25)_8$

Step 1:

We multiply each digit of **152.25** with its respective positional weight, and last we add the products of all the bits with its weight.

$\mathbf{(152.25)_8}=(1\times8^2)+(5\times8^1)+(2\times8^0)+(2\times8^{-1})+(5\times8^{-2})$

$\mathbf{(152.25)_8}=64+40+2+(2\times⅛)+(5\times1/64)$

$\mathbf{(152.25)_8}=64+40+2+0.25+0.078125$

$\mathbf{(152.25)_8}=106.328125$

So, the decimal number of the octal number 152.25 is **106.328125**

Octal to Binary Conversion

The process of converting octal to binary is the reverse process of binary to octal. We write the three bits binary code of each octal number digit.

Example 1: $(152.25)_8$

We write the three-bit binary digit for 1, 5, 2, and 5.

$\mathbf{(152.25)_8}=(001101010.010101)_2$

So, the binary number of the octal number 152.25 is $\mathbf{(001101010.010101)_2}$

Octal to hexadecimal conversion

For converting octal to hexadecimal, there are two steps required to perform, which are as follows:

1. In the first step, we will find the binary equivalent of number **25**.
2. Next, we have to make the pairs of four bits on both sides of the binary point. If there will be one, two, or three bits left in a pair of four bits pair, we add the required number of zeros on extreme sides and write the hexadecimal digits corresponding to each pair.

Example 1: $(152.25)_8$

Step 1:

We write the three-bit binary digit for 1, 5, 2, and 5.

$(152.25)_8$=$(001101010.010101)_2$

So, the binary number of the octal num

Step 2:

1. Now, we make pairs of four bits on both sides of the binary point.

0 0110 1010.0101 01

On the left side of the binary point, the first pair has only one digit, and on the right side, the last pair has only two-digit. To make them complete pairs of four bits, add zeros on extreme sides.

0000 0110 1010.0101 0100

2. Now, we write the hexadecimal digits, which correspond to each pair.

$(0000\ 0110\ 1010.0101\ 0100)_2$=$(6A.54)_{16}$

Hexa-decimal to other Number System

Like binary, decimal, and octal, hexadecimal numbers can also be converted into other number systems. The process of converting hexadecimal to decimal differs from the remaining one.

Hexa-decimal to Decimal Conversion

The process of converting hexadecimal to decimal is the same as binary to decimal. The process starts from multiplying the digits of hexadecimal numbers with its corresponding positional weights. And lastly, we add all those products.

Example 1: $(152A.25)_{16}$

Step 1:

We multiply each digit of **152A.25** with its respective positional weight, and last we add the products of all the bits with its weight.

$\mathbf{(152A.25)_{16}}=(1\times16^{3})+(5\times16^{2})+(2\times16^{1})+(A\times16^{0})+(2\times16^{-1})+(5\times16^{-2})$

$\mathbf{(152A.25)_{16}}=(1\times4096)+(5\times256)+(2\times16)+(10\times1)+(2\times16^{-1})+(5\times16^{-2})$

$\mathbf{(152A.25)_{16}}=4096+1280+32+10+(2\times1/16)+(5\times1/256)$

$\mathbf{(152A.25)_{16}}=5418+0.125+0.125$

$\mathbf{(152A.25)_{16}}=5418.14453125$

So, the decimal number of the hexadecimal number 152A.25 is **5418.14453125**

Hexadecimal to Binary Conversion

The process of converting hexadecimal to binary is the reverse process of binary to hexadecimal. We write the four bits binary code of each hexadecimal number digit.

Example 1: $(152A.25)_{16}$

We write the four-bit binary digit for 1, 5, A, 2, and 5.

$(152A.25)_{16} = (0001\ 0101\ 0010\ 1010.0010\ 0101)_2$

So, the binary number of the hexadecimal number 152.25 is **$(1010100101010.00100101)_2$**

Hexadecimal to Octal Conversion

For converting hexadecimal to octal, there are two steps required to perform, which are as follows:

1. In the first step, we will find the binary equivalent of the hexadecimal number.
2. Next, we have to make the pairs of three bits on both sides of the binary point. If there will be one or two bits left in a pair of three bits pair, we add the required number of zeros on extreme sides and write the octal digits corresponding to each pair.

Example 1: $(152A.25)_{16}$

Step 1:

We write the four-bit binary digit for 1, 5, 2, A, and 5.

$\mathbf{(152A.25)_{16}=(0001\ 0101\ 0010\ 1010.0010\ 0101)_2}$

So, the binary number of hexadecimal number 152A.25 is $\mathbf{(0011010101010.010101)_2}$

Step 2:

3. Then, we make pairs of three bits on both sides of the binary point.

001 010 100 101 010.001 001 010

4. Then, we write the octal digit, which corresponds to each pair.

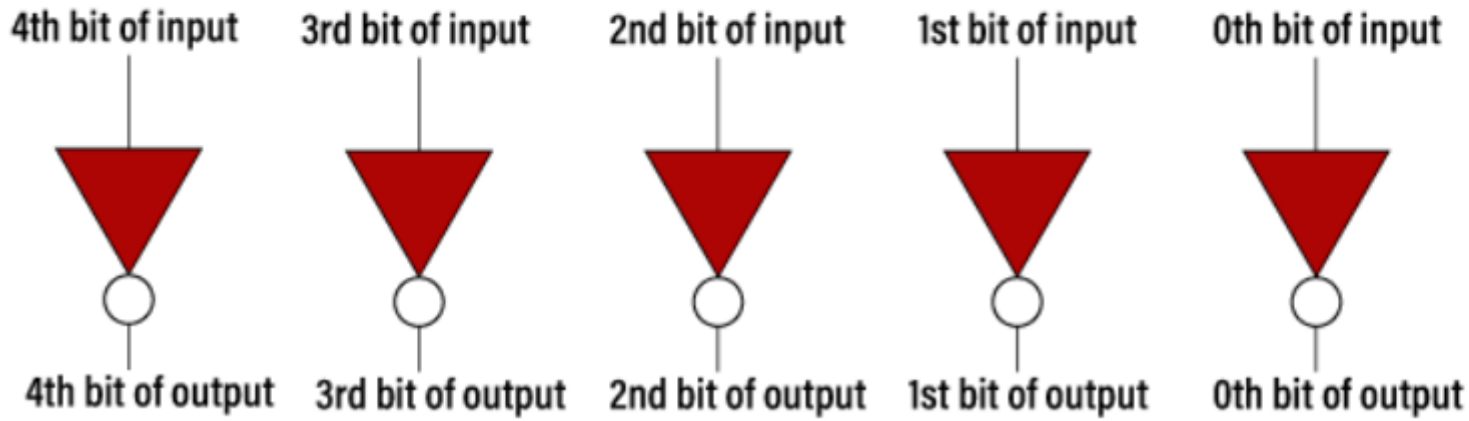
$\mathbf{(0010101001010.001001010)_2=(12452.112)_8}$

So, the octal number of the hexadecimal number 152A.25 is **12452.112**

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**.

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.

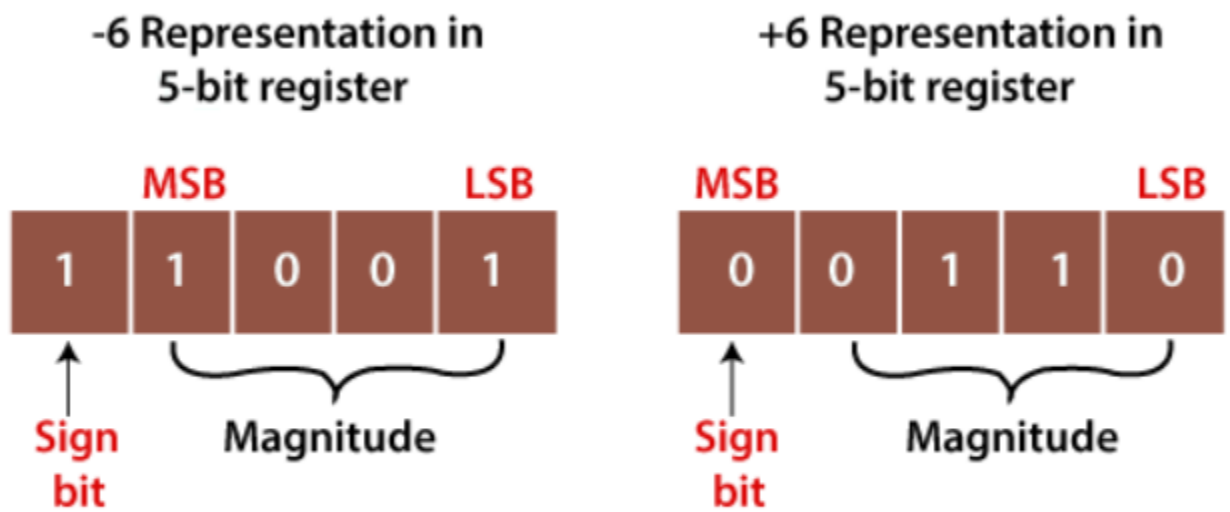
Example 1: +6 and -6

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.

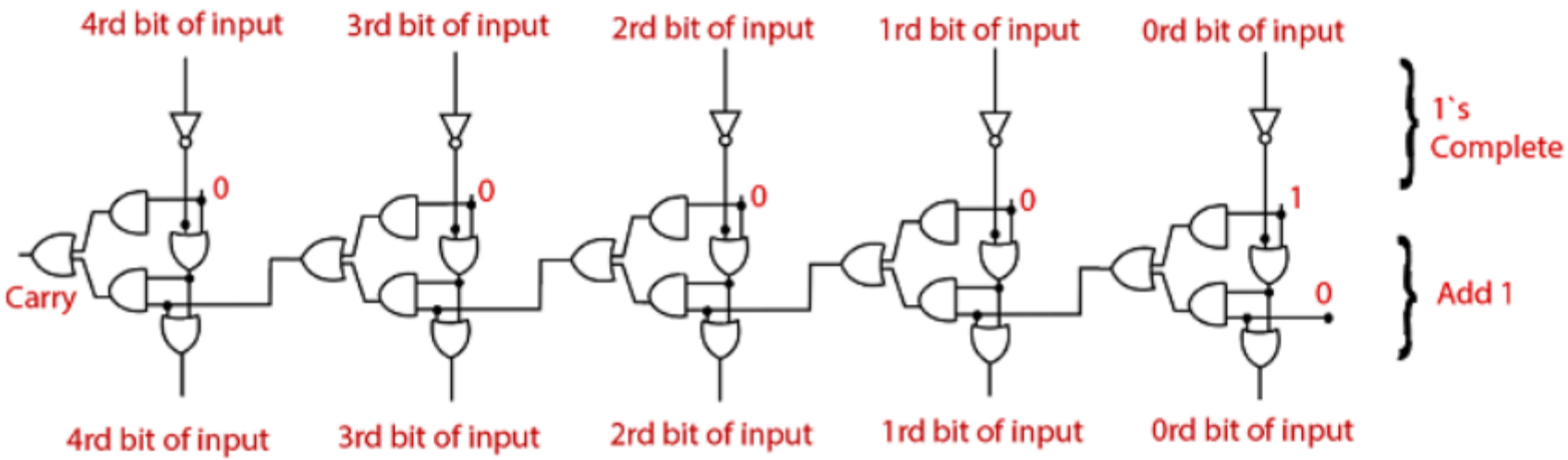
The -120 is represented in the 8-bit register in the following way:

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.

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.

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.

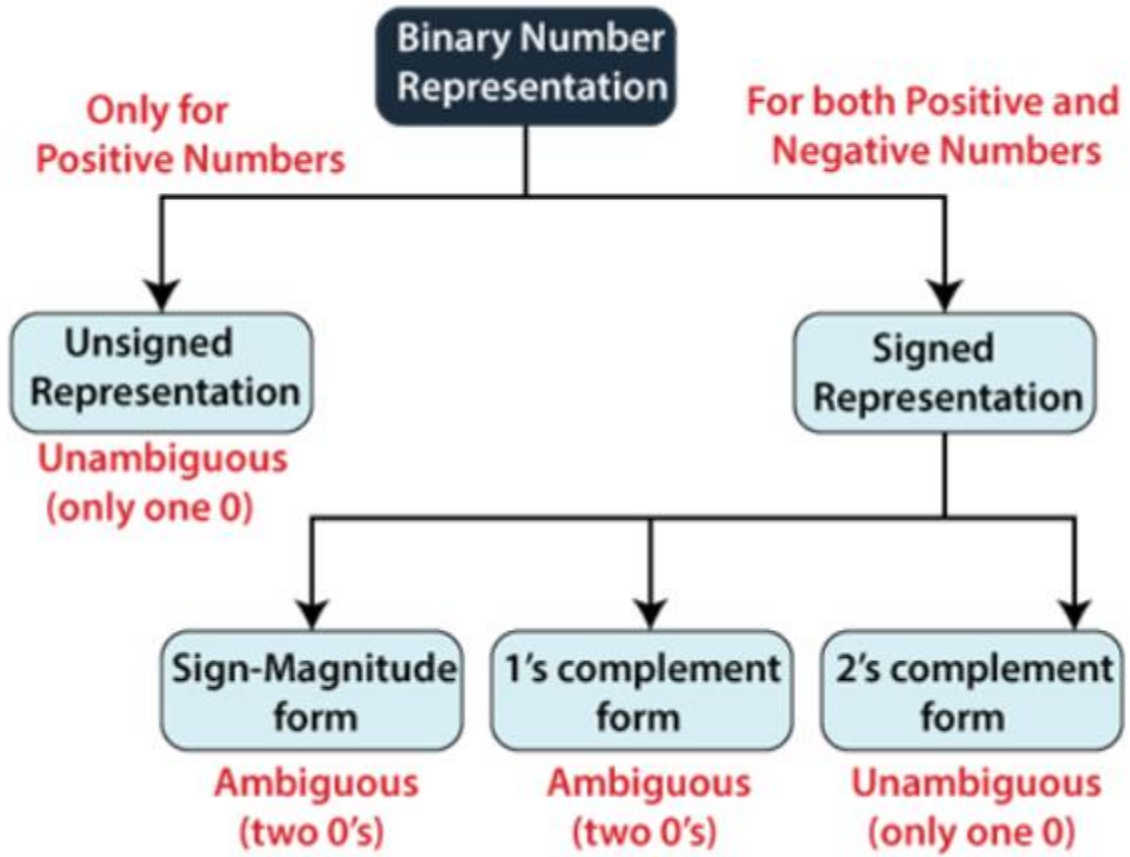
Binary Numbers Representation

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.

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 (2^n-1).


Binary Number Representation
Only for Positive Numbers
For both Positive and Negative Numbers
Unsigned Representation
Unambiguous (only one 0)
Signed Representation
Sign-Magnitude form
Ambiguous (two 0's)
1's complement form
Ambiguous (two 0's)
2's complement form
Unambiguous (only one 0)

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.

2. **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.

3. **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.