KTU NOTES

The learning companion.

KTU STUDY MATERIALS | SYLLABUS | LIVE NOTIFICATIONS | SOLVED QUESTION PAPERS

Module I

Number systems, Operations & Codes

Decimal, Binary, Octal and Hexadecimal Number Systems- Number Base Conversions. Addition, Subtraction, Multiplication and Division of binary numbers. Representation of negative numbers- Complements, Subtraction with complements. Addition and subtraction of BCD, Octal and Hexadecimal numbers. Binary codes- Decimal codes, Error detection codes, Reflected code, Character coding schemes – ASCII, EBCDIC.

Number Systems

If base or radix of a number system is 'r', then the numbers present in that number system are ranging from zero to r-1. The total numbers present in that number system is 'r'.

The following number systems are most commonly used.

- Decimal Number system
- Binary Number system
- Octal Number system
- Hexadecimal Number system

Decimal Number System

- The base or radix of Decimal number system is 10.
- numbers ranging from 0 to 9 are used.
- The part of the number that lies to the left of the decimal point is known as integer part.
- the part of the number that lies to the right of the decimal point is known as fractional part.

Binary Number System

- All digital circuits and systems use this binary number system.
- The base or radix of this number system is 2.
- the numbers 0 and 1 are used in this number system.
- The part of the number, which lies to the left of the binary point is known as integer part.
- the part of the number, which lies to the right of the binary point is known as fractional part.

Octal Number System

- The base or radix of octal number system is 8.
- numbers ranging from 0 to 7 are used in this number system.
- The part of the number that lies to the left of the octal point is known as integer

part.

- the part of the number that lies to the right of the octal point is known as fractional part.

Hexadecimal Number System

- The base or radix of Hexa-decimal number system is 16.
- numbers ranging from 0 to 9 and the letters from A to F are used in this number system.
- The decimal equivalent of Hexa-decimal digits from A to F are 10 to 15.
- The part of the number, which lies to the left of the hexadecimal point is known as integer part.
- the part of the number, which lies to the right of the Hexa decimal point is known as fractional part.

Equivalent Decimal, Binary, Octal and Hexadecimal Numbers

Decimal	Binary	Octal	Hexadecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Number Base Conversions

1. Decimal to Other Bases Conversion

1.1 Decimal to Binary Conversion

The following two types of operations take place, while converting decimal number into its equivalent binary number.

- Division of integer part and successive quotients with base 2.
- Multiplication of fractional part and successive fractions with base

Example

Consider the decimal number 58.25. Here, the integer part is 58 and fractional part is 0.25.

Step 1 − Division of 58 and successive quotients with base 2.

Operation	Quotient	Remainder
58/2	29	**0** LSB
29/2	14	**1**
14/2	7	**0**
7/2	3	**1**
3/2	1	**1**
½	0	**1** MSB

$\Rightarrow 58_{10} = 111010_2$

Therefore, the integer part of equivalent binary number is **111010**.

Step 2 – Multiplication of 0.25 and successive fractions with base 2.

Operation	Result	Carry
0.25 x 2	0.5	0
0.5 x 2	1.0	1
0 X 2	0	0

$\Rightarrow .25_{10} = .01_2$

Therefore, the fractional part of equivalent binary number is **.01**

$\Rightarrow 58.25_{10} = 111010.01_2$

Therefore, the binary equivalent of decimal number 58.25 is 111010.01

1.2 Decimal to Octal Conversion

The following two types of operations take place, while converting decimal number into its equivalent octal number.

- Division of integer part and successive quotients with base 8.
- Multiplication of fractional part and successive fractions with base 8.

Example
Consider the decimal number 58.25. Here, the integer part is 58 and fractional part is 0.25.

Step 1 − Division of 58 and successive quotients with base 8.

Operation	Quotient	Remainder
58/8	7	**2**
7/8	0	**7**

$\Rightarrow 58_{10} = 72_8$

Therefore, the integer part of equivalent octal number is **72**.

Step 2 − Multiplication of 0.25 and successive fractions with base 8.

Operation	Result	Carry
0.25 x 8	2.00	2
-	0.00	-

$\Rightarrow ..25_{10} = .2_8$

Therefore, the fractional part of equivalent octal number is .2

$\Rightarrow 58.25_{10} = 72.2_8$
Therefore, the octal equivalent of decimal number 58.25 is 72.2.

1.3 Decimal to Hexa-Decimal Conversion

The following two types of operations take place, while converting decimal number into its equivalent hexa-decimal number.

- Division of integer part and successive quotients with base 16.
- Multiplication of fractional part and successive fractions with base 16.

Example
Consider the decimal number 58.25. Here, the integer part is 58 and decimal part is 0.25.

Step 1 − Division of 58 and successive quotients with base 16.

Operation	Quotient	Remainder
58/16	3	10=A
3/16	0	3

$\Rightarrow 58_{10} = 3A_{16}$

Therefore, the **integer part** of equivalent Hexa-decimal number is 3A.

Step 2 − Multiplication of 0.25 and successive fractions with base 16.

Operation	Result	Carry
0.25 x 16	4.00	4
-	0.00	-

$\Rightarrow .25_{10} = .4_{16}$

Therefore, the fractional part of equivalent Hexa-decimal number is .4.

$\Rightarrow 58.25_{10} = 3A.4_{16}$

Therefore, the Hexa-decimal equivalent of decimal number 58.25 is 3A.4.

2. Binary Number to other Bases Conversion

2.1 Binary to Decimal Conversion

For converting a binary number into its equivalent decimal number, first multiply the bits of binary number with the respective positional weights and then add all those products.

Example

Consider the binary number 1101.11.

Mathematically, we can write it as

$1101.11_2 = (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) + (1 \times 2^{-1}) + (1 \times 2^{-2})$

$\Rightarrow 1101.11_2 = 8 + 4 + 0 + 1 + 0.5 + 0.25 = 13.75$

$\Rightarrow 1101.11_2 = 13.75_{10}$

Therefore, the decimal equivalent of binary number 1101.11 is 13.75.

2.2 Binary to Octal Conversion

We know that the bases of binary and octal number systems are 2 and 8 respectively. Three bits of binary number is equivalent to one octal digit, since $2^3 = 8$.

Follow these two steps for converting a binary number into its equivalent octal number.

- Start from the binary point and make the groups of 3 bits on both sides of binary point. If one or two bits are less while making the group of 3 bits, then include required number of zeros on extreme sides.
- Write the octal digits corresponding to each group of 3 bits.

Example

Consider the binary number 101110.01101.

Step 1 − Make the groups of 3 bits on both sides of binary point. 101 110.011 01

Here, on right side of binary point, the last group is having only 2 bits. So, include one zero on extreme side in order to make it as group of 3 bits.

$$\Rightarrow 101\ 110.011\ 010$$

Step 2 − Write the octal digits corresponding to each group of 3 bits. $\Rightarrow 101110.011010_2 = 56.32_8$

Therefore, the octal equivalent of binary number 101110.01101 is 56.32.

2.3 Binary to Hexa-Decimal Conversion

We know that the bases of binary and Hexa-decimal number systems are 2 and 16 respectively. Four bits of binary number is equivalent to one Hexa-decimal digit, since $2^4 = 16$.

Follow these two steps for converting a binary number into its equivalent Hexa decimal number.

- Start from the binary point and make the groups of 4 bits on both sides of binary point. If some bits are less while making the group of 4 bits, then include required number of zeros on extreme sides.
- Write the Hexa-decimal digits corresponding to each group of 4 bits.

Example

Consider the binary number 101110.01101

Step 1 − Make the groups of 4 bits on both sides of binary point. 10 1110.0110 1

Here, the first group is having only 2 bits. So, include two zeros on extreme side in order to make it as group of 4 bits. Similarly, include three zeros on extreme side in order to make the last group also as group of 4 bits.

$$\Rightarrow 0010\ 1110.0110\ 1000$$

Step 2 − Write the Hexa-decimal digits corresponding to each group of 4 bits.

$$\Rightarrow 00101110.01101000_2 = 2E.68_{16}$$

Therefore, the Hexa-decimal equivalent of binary number 101110.01101 is 2E.68.

3. Octal Number to other Bases Conversion

3.1 Octal to Decimal Conversion

For converting an octal number into its equivalent decimal number, first multiply the digits of octal number with the respective positional weights and then add all those products.

Example

Consider the octal number 145.23.

Mathematically, we can write it as

$145.23_8 = (1 \times 8^2) + (4 \times 8^1) + (5 \times 8^0) + (2 \times 8^{-1}) + (3 \times 8^{-2})$
$\Rightarrow 145.23_8 = 64 + 32 + 5 + 0.25 + 0.05 = 101.3$

$\Rightarrow 145.23_8 = 101.3_{10}$

Therefore, the decimal equivalent of octal number 145.23 is 101.3.

3.2 Octal to Binary Conversion

The process of converting an octal number to an equivalent binary number is just opposite to that of binary to octal conversion. By representing each octal digit with 3 bits, we will get the equivalent binary number.

Example

Consider the octal number 145.23.

Represent each octal digit with 3 bits.

$$145.23_8 = 001100101.010011_2$$

The value doesn't change by removing the zeros, which are on the extreme side. $\Rightarrow$

$$145.23_8 = 1100101.010011_2$$

Therefore, the binary equivalent of octal number 145.23 is 1100101.010011.

3.3 Octal to Hexa-Decimal Conversion

Follow these two steps for converting an octal number into its equivalent Hexa decimal number.

- Convert octal number into its equivalent binary number.
- Convert the above binary number into its equivalent Hexa-decimal number.

Example

Consider the octal number 145.23

In previous example, we got the binary equivalent of octal number 145.23 as 1100101.010011.

By following the procedure of binary to Hexa-decimal conversion, we will get

$$1100101.010011_2 = 65.4C_{16}$$

$$\Rightarrow 145.23_8 = 65.4C_{16}$$

Therefore, the Hexa-decimal equivalent of octal number 145.23 is 65.4*C*.

4. Hexa-Decimal Number to other Bases Conversion

4.1 Hexa-Decimal to Decimal Conversion

For converting Hexa-decimal number into its equivalent decimal number, first multiply the digits of Hexa-decimal number with the respective positional weights and then add all those products.

Example

Consider the Hexa-decimal number 1A5.2

Mathematically, we can write it as

$$1A5.2_{16} = (1 \times 16^2) + (10 \times 16^1) + (5 \times 16^0) + (2 \times 16^{-1})$$

$\Rightarrow 1A5.2_{16} = 256 + 160 + 5 + 0.125 = 421.125$

$\Rightarrow 1A5.2_{16} = 421.125_{10}$

Therefore, the decimal equivalent of Hexa-decimal number 1A5.2 is 421.125.

4.2 Hexa-Decimal to Binary Conversion

The process of converting Hexa-decimal number into its equivalent binary number is just opposite to that of binary to Hexa-decimal conversion. By representing each Hexa-decimal digit with 4 bits, we will get the equivalent binary number.

Example

Consider the Hexa-decimal number 65.4C

Represent each Hexa-decimal digit with 4 bits.

$$65.4C_{16} = 01100101.01001100_2$$

The value doesn't change by removing the zeros, which are at two extreme sides. $\Rightarrow$ $65.4C_{16} = 1100101.010011_2$
Therefore, the binary equivalent of Hexa-decimal number 65.4C is 1100101.010011.

4.3 Hexa-Decimal to Octal Conversion

Follow these two steps for converting Hexa-decimal number into its equivalent octal number.

- Convert Hexa-decimal number into its equivalent binary number.
- Convert the above binary number into its equivalent octal number.

Example

Consider the Hexa-decimal number 65.4C

In previous example, we got the binary equivalent of Hexa-decimal number 65.4C as 1100101.010011.
By following the procedure of binary to octal conversion, we will get

$$1100101.010011_2 = 145.23_8$$
$$\Rightarrow 65.4C_{16} = 145.23_8$$

Therefore, the octal equivalent of Hexa-decimal number 65.4*C* is 145.23.

Binary Arithmetic

Binary Addition

Four rules of binary addition.

Case	A + B	Sum	Carry
1	0 + 0	0	0
2	0 + 1	1	0
3	1 + 0	1	0
4	1 + 1	0	1

In fourth case, a binary addition is creating a sum of (1 + 1 = 10) i.e. 0 is written as sum column and 1 as carry.

Example

0011010 + 001100 = 00100110

```
  1 1          carry
  0011010  = 26(10)
+ 0001100  = 12(10)
-------
  0100110  = 38(10)
```

Binary Subtraction

There are four rules of binary subtraction.

Case	A - B	Subtract	Borrow
1	0 - 0	0	0
2	1 - 0	1	0
3	1 - 1	0	0
4	0 - 1	0	1

In fourth case, 0 - 1 requires borrow of 1. After borrowing 1, 0 becomes 10. So 10 – 1, i.e. 2-1 = 1, hence difference is 1 and borrow is 1

Example

0011010 - 001100 = 00001110

```
    1 1      borrow
  0 0 1 1 0 1 0   = 26₁₀
- 0 0 0 1 1 0 0   = 12₁₀
-----------------
  0 0 0 1 1 1 0   = 14₁₀
```

Binary Multiplication

Case	A x B	Multiplication
1	0 x 0	0
2	0 x 1	0
3	1 x 0	0
4	1 x 1	1

Example:

0011010 x 001100 = 100111000

```
    0 0 1 1 0 1 0   = 26₁₀
x   0 0 0 1 1 0 0   = 12₁₀
-----------------
    0 0 0 0 0 0 0
  0 0 0 0 0 0 0
0 0 1 1 0 1 0
0 0 1 1 0 1 0
-----------------
0 1 0 0 1 1 1 0 0 0   = 312₁₀
```

Example: Using the binary multiplication rules, multiply 110_2 and 11_2.

```
    1 1 0
  x   1 1
---------
  ① 1 1 0
  1 1 0 x
---------
1 0 0 1 0
```

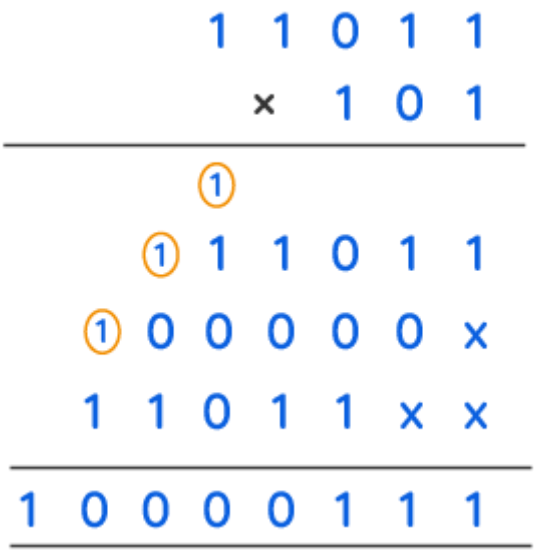
Therefore, the product of 110_2 and 11_2 is 10010_2.

Example: Using the binary multiplication rules, find the product of 11011_2 and 101_2.

Solution:

GIven multiplicand = 11011 and multiplier = 101

On multiplying we get,



Therefore, the product of 11011 and 101 is 10000111.

Binary Division

Binary division problems can be solved by using the long division method, which is one of the most efficient and easiest ways to divide binary numbers. These are the steps to be followed in a binary division operation:

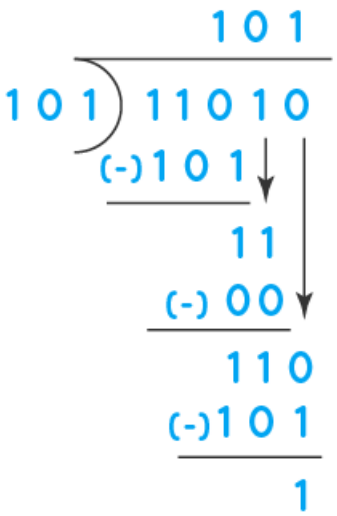
- **Step 1:** Compare the divisor with the dividend. If the divisor is larger, place 0 as the quotient, then bring the second bit of the dividend down. If the divisor is smaller, multiply it with 1 and the result becomes the subtrahend. Then, subtract the subtrahend from the minuend to get the remainder.
- **Step 2:** Then bring down the next number bit from the dividend portion and perform step 1 again.
- **Step 3:** Repeat the same process until the remainder becomes zero or the whole dividend is divided.

Example: Consider two binary numbers, B = 011010 and C = 0101. Divide B by C.

Solution

Given: Dividend, 011010 and the divisor,C = 0101.

Step1: Since the zero in the most significant bit position doesn't change the value of the number, let's remove it from both the dividend and divisor. So the dividend becomes 11010, and the divisor becomes 101.



Step 2: Let us use the long-division method. In this step, compare the divisor 101 with the first digit in the dividend 11010, since the divisor is smaller, it will be multiplied with 1 and the result will be the subtrahend.

As per the binary multiplication rules:

- 1 × 1 = 1
- 1 × 0 = 0
- 0 × 1 = 0
- 0 × 0 = 0

So, 101 × 1 = 101, and this result is written below.

Step 3: Subtract the subtrahend 101 from the minuend 110.

As per the binary subtraction rules,

- 0 - 1 = 1, we need to borrow 1 from the next more significant bit.
- 0 - 0 = 0
- 1 - 1 = 0
- 1 - 0 = 0

When we apply the above rules, this is how the calculation is done:

- For the first digit on the right, we have to subtract (0 - 1). So, we borrow a 1 from the digit on the left or the next higher order digit. Therefore, the result is 1.

- Then, (0 - 0 = 0) since the number in the next higher order digit becomes 0 after borrowing.
- 1 - 1 = 0 in the second next higher order digit.
- So, 110 - 101 = 001, and this result is written below.

Step 4: As per the rules of division, the next least significant bit comes down, and the divisor is multiplied by 1. Since the result, 101 is bigger than the minuend 0011, this step cannot be completed. Then, we have to go to the next step

Step 5: We write 0 as the next bit of the quotient and then, the least significant bit 0 comes down.

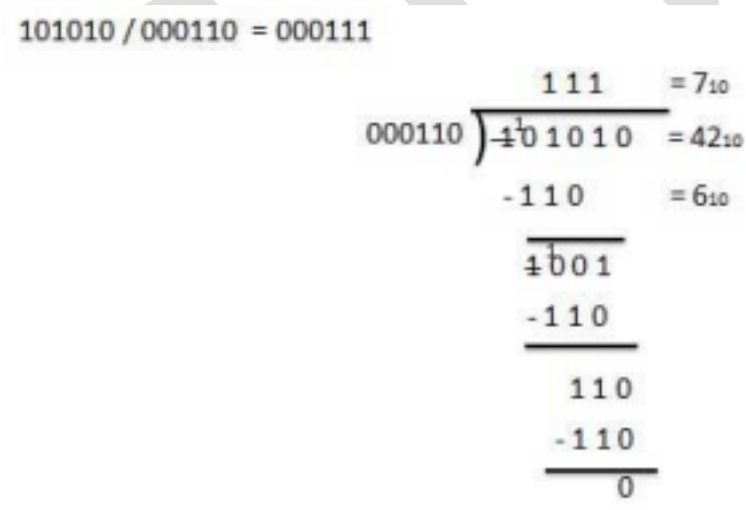
Step 6: Again the divisor is multiplied by 1 and the result is written as 101 × 1 = 101.

Step 7: Now we are at the final step. As per the binary subtraction, we subtract 101 from 110. We get, 110 - 101 = 001. The remainder is similar to Step 3, as all the numbers are the same.

The binary division operation is completed now and we get the following result.

- Quotient = 101
- Remainder = 001 = 1

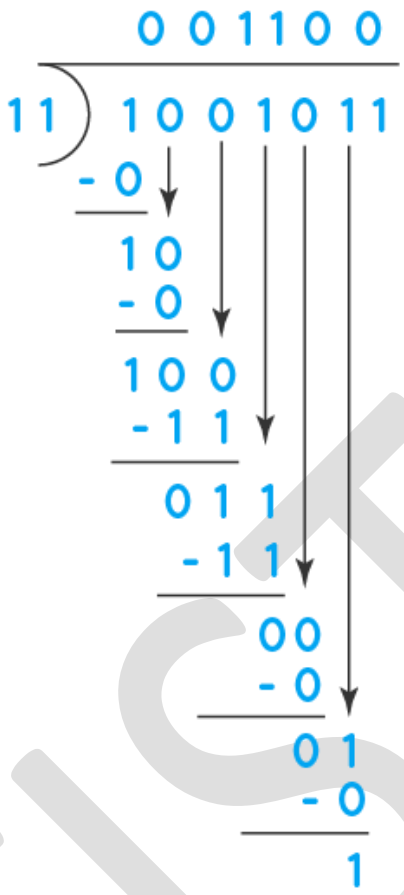
Example: Evaluate $101010_2 \div 110_2$ using the long-division method.



Example: Evaluate $1001011_2 \div 11_2$ using the long-division method.

Solution:

Here, Dividend = 1001011, Divisor = 11.

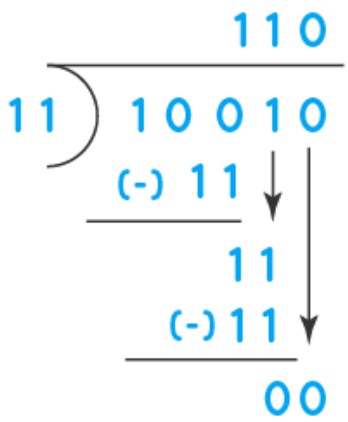


Answer: Quotient = 1100_2, Remainder = 1

Example: Evaluate $10010_2 \div 11_2$ using the long-division method.

Solution:

Here, Dividend = 100102, Divisor = 112



Answer: Quotient = 110_2, Remainder = 0

Complements

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

Diminished Radix Complement or (r-1)'s Complement

Given a number N in base r having n digits,

the (r–1)'s complement of N is defined as: $(r^n - 1) - N$

Example for 6-digit decimal numbers:

- 9's complement is $(r^n - 1) - N = (10^6 - 1) - N = 999999 - N$
- 9's complement of 546700 is 999999–546700 = 453299

Example for 7-digit binary numbers:

- 1's complement is $(r^n - 1) - N = (2^7 - 1) - N = 1111111 - N$
- 1's complement of 1011000 is 1111111–1011000 = 0100111

Radix Complement or r's complement

The r's complement of an n-digit number N in base r is defined as $r^n - N$ for $N \neq 0$ and as 0 for N = 0.

Comparing with the (r – 1) 's complement, we note that the r's complement is obtained by adding 1 to the (r – 1) 's complement, since $r^n - N = [(r^n - 1) - N] + 1$.

Example 1: $(1011000)_2$

This number has a base of 2, which means it is a binary number. So, for the binary numbers, the value of r is 2, and r-1 is 2-1=1. So, we can calculate the 1's and 2's complement of the number.

1's complement of the number 1011000 is calculated as:

$=\{(2^7)_{10}-1\}-(1011000)_2$
$=\{(128)_{10}-1\}-(1011000)_2$
$=\{(127)_{10}\}-(1011000)_2$
$=1111111_2-1011000_2$
$=0100111$

2's complement of the number 1011000 is calculated as:

$=(2^7)_{10}-(1011000)_2$
$=(128)_{10}-(1011000)_2$
$=10000000_2-1011000_2$
$=0101000_2$

Example 2: $(155)_{10}$

This number has a base of 10, which means it is a decimal number. So, for the decimal numbers, the value of r is 10, and r-1 is 10-1=9. So, we can calculate the 10's and 9's complement of the number.

9's complement of the number 155 is calculated as:

$=\{(10^3)_{10}-1\}-(155)_{10}$
$=(1000-1)-155$
$=999-155$
$=(844)_{10}$

10's complement of the number 1011000 is calculated as:

$=(10^3)_{10}-(155_{10}$
$=1000-155$
$=(845)_{10}$

Example 3: $(172)_8$

This number has a base of 8, which means it is an octal number. So, for the octal numbers, the value of r is 8, and r-1 is 8-1=7. So, we can calculate the 8's and 7's complement of the number.

7's complement of the number 172 is calculated as:

$=\{(8^3)_{10}-1\}-(172)_8$
$=((512)_{10}-1)-(132)_8$

$=(511)_{10}-(122)_{10}$
$=(389)_{10}$
$=(605)_8$

8's complement of the number 172 is calculated as:

$=(8^3)_{10}-(172)_8$
$=(512)_{10}-172_8$
$=512_{10}-122_{10}$
$=390_{10}$
$=606_8$

Example 4: $(F9)_{16}$

This number has a base of 16, which means it is a hexadecimal number. So, for the hexadecimal numbers, the value of r is 16, and r-1 is 16-1=15. So, we can calculate the 16's and 15's complement of the number.

15's complement of the number F9 is calculated as:

$\{(16^2)_{10}-1\}-(F9)_{16}$
$(256-1)_{10}-F9_{16}$
$255_{10}-249_{10}$
$(6)_{10}$
$(6)_{16}$

16's complement of the number F9 is calculated as:

$\{(16^2)_{10}\}-(F9)_{16}$
$256_{10}-249_{10}$
$(7)_{10}$
$(7)_{16}$

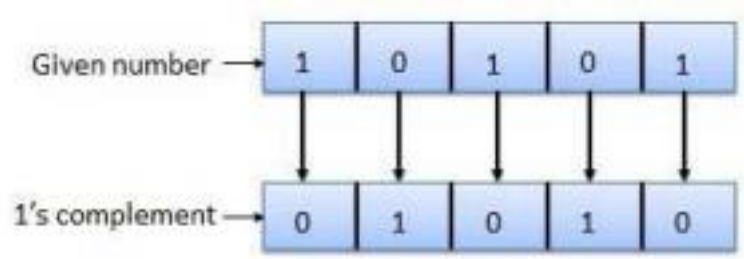
Binary system complements

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

1’s complement: is obtained by subtracting each digit in the number from 1 or by simply inverting the digits.

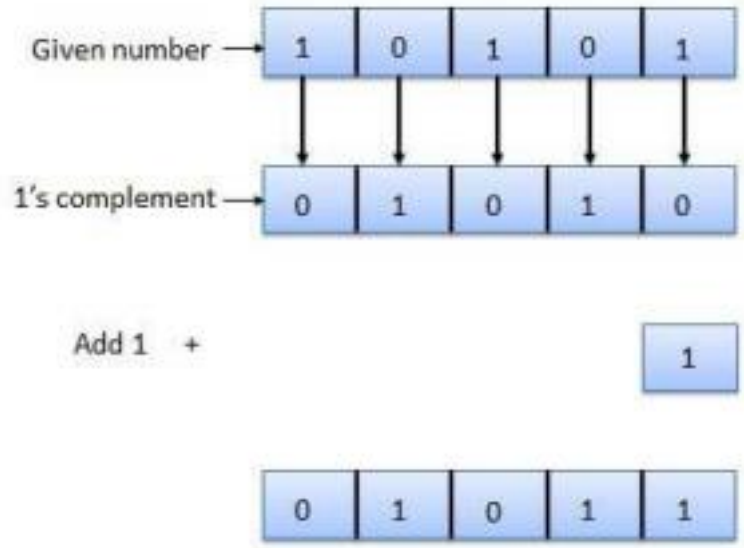
The 1's complement of a number is found by changing all 1's to 0's and all 0's to 1's. This is called as taking complement or 1's complement. Example of 1's Complement is as follows.



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.



Decimal System Complements

• 9's complement: is obtained by subtracting each digit in the number from 9 • 10's complement: is obtained by adding 1 to 9's complement

Octal System Complements

• 7's complement: is obtained by subtracting each digit in the number from 7 • 8's complement: is obtained by adding 1 to 7's complement

Hexadecimal System Complements

- 15's complement: is obtained by subtracting each digit in the number from 15(F)
- 16's complement: is obtained by adding 1 to 15's complement

Binary Numbers Representation

Representation of Un-Signed Binary Numbers

The bits present in the un-signed binary number holds the magnitude of a number. That means, if the un-signed binary number contains 'N' bits, then all N bits represent the magnitude of the number, since it doesn't have any sign bit.

Example

Consider the decimal number 108. The binary equivalent of this number is 1101100. This is the representation of unsigned binary number.

$$108_{10} = 1101100_2$$

It is having 7 bits. These 7 bits represent the magnitude of the number 108.

Representation of Signed Binary Numbers

The Most Significant Bit MSB of signed binary numbers is used to indicate the sign of the numbers. Hence, it is also called as sign bit. The positive sign is represented by placing '0' in the sign bit. Similarly, the negative sign is represented by placing '1' in the sign bit.
If the signed binary number contains 'N' bits, then N−1 bits only represent the magnitude of the number since one-bit MSB is reserved for representing sign of the number.

There are three types of representations for signed binary numbers

- Sign-Magnitude form
- 1's complement form
- 2's complement form

Representation of a positive number in all these 3 forms is same. But, only the representation of negative number will differ in each form.

Example

Consider the positive decimal number +108. The binary equivalent of magnitude of this number is 1101100. These 7 bits represent the magnitude of the number 108. Since it is positive number, consider the sign bit as zero, which is placed on left most side of magnitude.

$$+108_{10} = 01101100_2$$

Therefore, the signed binary representation of positive decimal number +108 is 01101100. So, the same representation is valid in sign-magnitude form, 1's complement form and 2's complement form for positive decimal number +108.

Sign-Magnitude form

In sign-magnitude form, the MSB is used for representing sign of the number and the remaining bits represent the magnitude of the number. So, just include sign bit at the left most side of unsigned binary number. This representation is similar to the signed decimal numbers representation.

For an n-bit number representation:

– The most significant bit (MSB) indicates sign (0: positive, 1: negative).

– The remaining (n-1) bits represent the magnitude of the number.

Decimal	Signed magnitude representation in 4 bits	Decimal	Signed magnitude representation in 4 bits
+0	0000	-0	1000
+1	0001	-1	1001
+2	0010	-2	1010
+3	0011	-3	1011
+4	0100	-4	1100
+5	0101	-5	1101
+6	0110	-6	1110
+7	0111	-7	1111

Note: A problem- Two different representations for zero.

Example

Consider the negative decimal number -108. The magnitude of this number is 108. We know the unsigned binary representation of 108 is 1101100. It is having 7 bits. All these bits represent the magnitude.

Since the given number is negative, consider the sign bit as one, which is placed on left most side of magnitude.

$$-108_{10} = 11101100_2$$

Therefore, the sign-magnitude representation of -108 is 11101100.

1's complement form

The 1's complement of a number is obtained by complementing all the bits of signed binary number. So, 1's complement of positive number gives a negative number. Similarly, 1's complement of negative number gives a positive number.

That means, if you perform two times 1's complement of a binary number including sign bit, then you will get the original signed binary number.

Basic idea:

– Positive numbers are represented exactly as in sign-magnitude form.

– Negative numbers are represented in 1's complement form. •

How to compute the 1's complement of a number?

– Complement every bit of the number (1 to 0, and 0 to 1).

– Most Significant Bit (MSB) will indicate the sign of the number (0: positive, 1: negative).

Decimal	1's complement representation in 4 bits	Decimal	1's complement representation in 4 bits
+0	0000	-0	1111
+1	0001	-1	1110
+2	0010	-2	1101
+3	0011	-3	1100
+4	0100	-4	1011
+5	0101	-5	1010
+6	0110	-6	1001
+7	0111	-7	1000

Note: A problem- Two different representations for zero.

Advantage of 1's complement representation:

– Subtraction can be done using addition.

– Leads to substantial saving in circuitry.

Example

Consider the negative decimal number -108. The magnitude of this number is 108. We know the signed binary representation of 108 is 01101100.

It is having 8 bits. The MSB of this number is zero, which indicates positive number. Complement of zero is one and vice-versa. So, replace zeros by ones and ones by zeros in order to get the negative number.

$$-108_{10} = 10010011_2$$

Therefore, the 1's complement of -108_{10} is 10010011_{2}.

2's complement form

The 2's complement of a binary number is obtained by adding one to the 1's complement of signed binary number. So, 2's complement of positive number gives a negative number. Similarly, 2's complement of negative number gives a positive number.

That means, if you perform two times 2's complement of a binary number including sign bit, then you will get the original signed binary number.

Basic idea:

– Positive numbers are represented exactly as in sign-magnitude form.

– Negative numbers are represented in 2's complement form.

How to compute the 2's complement of a number?

– Complement every bit of the number (1 to 0 and 0 to 1), and then add one to the resulting number.

– MSB will indicate the sign of the number (0: positive, 1: negative).

Decimal	2's complement representation in 4 bits	Decimal	2's complement representation in 4 bits
+0	0000	-0	-
+1	0001	-1	1111
+2	0010	-2	1110
+3	0011	-3	1101
+4	0100	-4	1100
+5	0101	-5	1011
+6	0110	-6	1010
+7	0111	-7	1001

Note: Unique representations for zero.

Advantage of 2' s complement representation:

– Unique representation of zero.

– Subtraction can be done using addition.

– Leads to substantial saving in circuitry.

Example

Consider the negative decimal number -108.

We know the 1's complement of $(-108)_{10}$ is $(10010011)_2$
2's compliment of 108_{10} = 1's compliment of 108_{10} + 1 =
10010011 + 1
= 10010100

Therefore, the 2's complement of 108_{10} is 10010100_2.

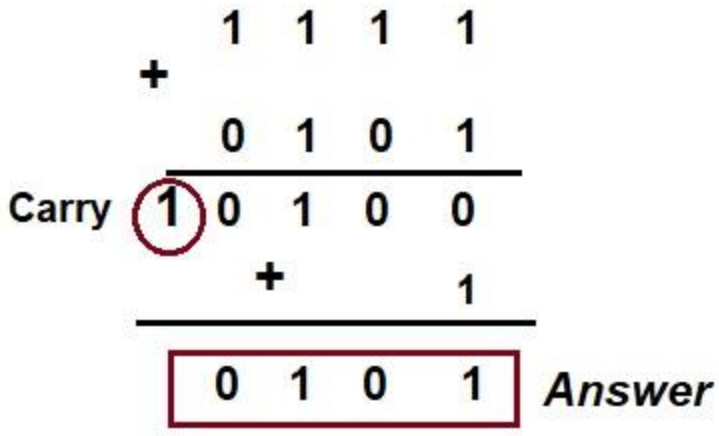
Subtraction with complements

1's Complement Subtraction

- **Subtraction of Smaller Number from Larger Number:**

The steps for 1's complement subtraction of a smaller number from a larger binary number are as follows:
Step-1: Determine the 1's complement of the smaller number.
Step-2: Add this to the larger number.
Step-3: Remove the carry and add it to the result. This carry is called end-around-carry.

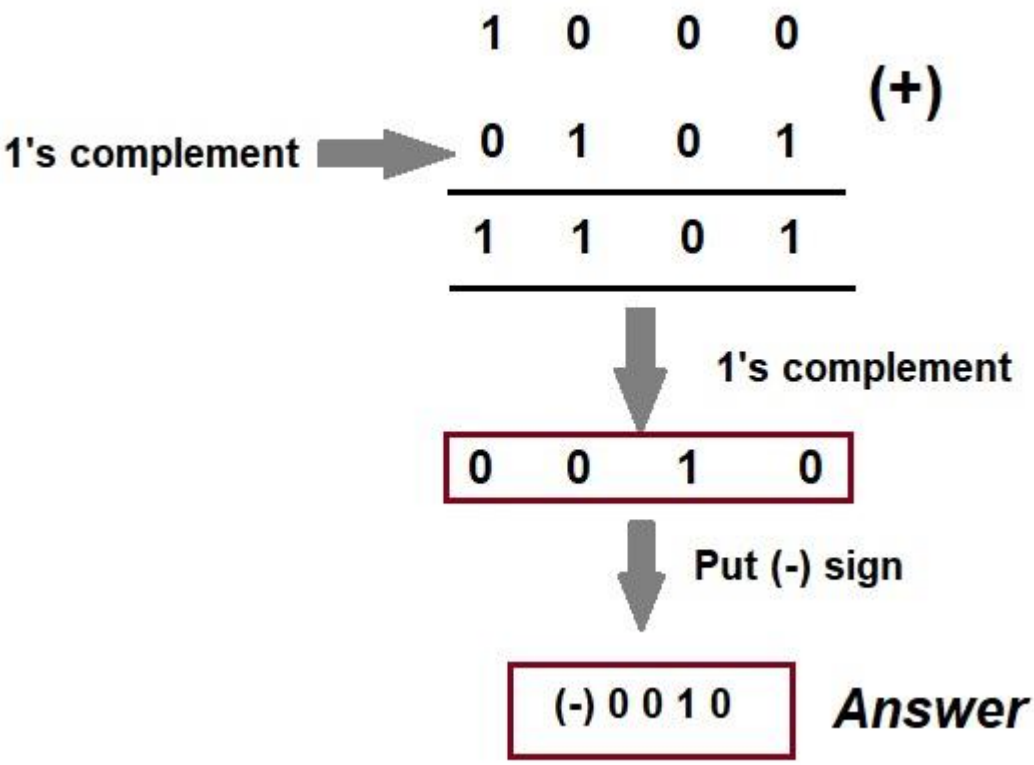
Example-1: Subtract $(1010)_2$ from $(1111)_2$ using 1's complement method.
Step-1: Find the 1's complement of 1010. It will be found by replacing all 0 to 1 and all 1 to 0. In this way, the required 1's complement will be 0101.
Step-2: In this step, we need to add the value calculated in step-1 to 1111. This is shown below.



Note: carry 1 has been added to the addition to get the subtraction of the binary numbers. Thus, the subtraction is done by actually adding two binary numbers.

- **Subtraction of Larger Number from Smaller Number:**

The steps involved in 1's complement subtraction of a larger number from a smaller number are as follows:
Step-1: Determine the 1's complement of the larger number.
Step-2: Add this to the smaller number.
Step-3: The answer is the 1's complement of the true result and opposite in sign. There is no carry.
Example: Subtract $(1010)_2$ from $(1000)_2$ using 1's complement method.
The 1's complement of $(1010)_2$ is $(0101)_2$. Now, we will add this with the smaller number and finally take 1's complement of the result to get the answer. This is shown below.



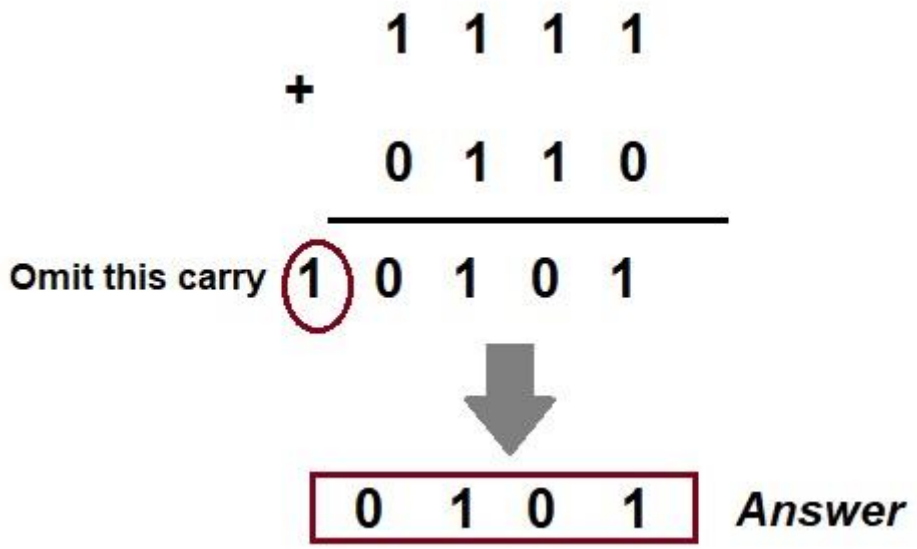
Note: no carry has been obtained while subtracting a larger number from a smaller number. Further, a minus sign has been put.

2's Complement Subtraction:

- **Subtraction of Smaller Number from Larger Number**

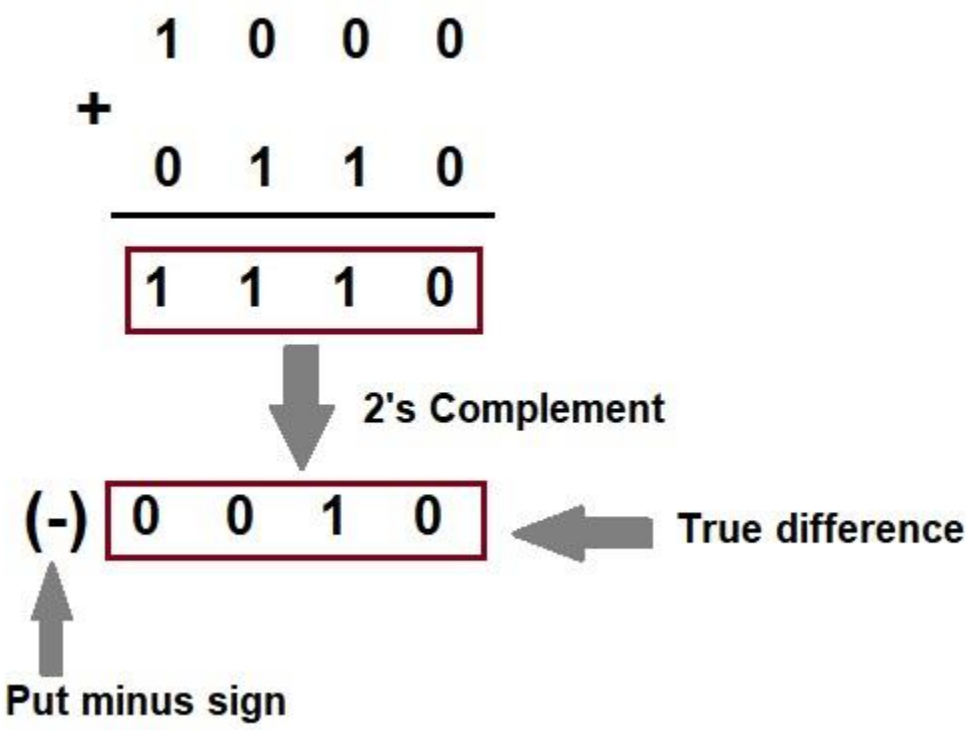
To subtract a smaller number from a larger number using 2's complement subtraction, following steps are to be followed:
Step-1: Determine the 2's complement of the smaller number

Step-2: Add this to the larger number.
Step-3: Omit the carry. Note that, there is always a carry in this case.
Following example illustrate the above mentioned steps:
Exampe-1: Subtract $(1010)_2$ from $(1111)_2$ using 2's complement method.
Solution:
Step-1: 2's complement of $(1010)_2$ is $(0110)_2$.
Step-2: Add $(0110)_2$ to $(1111)_2$. This is shown below.



- **Subtraction of Larger Number from Smaller Number:**

To subtract a larger number from a smaller number using 2's complement subtraction, following steps are to be followed:
Step-1: Determine the 2's complement of the smaller number
Step-2: Add this to the larger number.
Step-3: There is no carry in this case. The result is in 2's complement form and is negative.
Step-4: To get answer in true form, take 2's complement and change its sign.
xample-2: Subtract $(1010)_2$ from $(1000)_2$ using 2's complement.
Solution:
Step-1: Find the 2's complement of $(1010)_2$. It is $(0110)_2$.
Step-2: Add $(0110)_2$ to $(1000)_2$.



Advantages:
Though both 1's and 2's complement method for subtracting binary numbers seems to be complicated when compared with direct method of subtraction of two binary numbers, both have advantage when applied using logic circuits, because they allow subtraction to be done using only addition. The 1's and 2's complement of a binary number can easily be arrived at using logic circuits; the advantage in 2's complement subtraction is that the end-around-carry operation present in 1's complement method is not involved here.

BCD or **Binary Coded Decimal**

- BCD **or** Binary Coded Decimal **is** that number system or code which has the binary numbers or digits to represent a decimal number.
  In **BCD** we can use the binary number from 0000-1001 only, which are the decimal equivalent from 0-9 respectively.
- if a number have single decimal digit then it's equivalent Binary Coded Decimal will be the respective four binary digits of that decimal number
- if the number contains two decimal digits then it's equivalent BCD will be the respective eight binary of the given decimal number. Four for the first decimal digit and next four for the second decimal digit.
- Let, $(12)_{10}$ be the decimal number whose equivalent Binary coded decimal will be 00010010. Four bits from L.S.B is binary equivalent of 2 and next four is the binary equivalent of 1.

- For 0 to 9 decimal numbers both binary and BCD is equal but when decimal number is more than one bit BCD differs from binary.

Decimal number	Binary number	Binary Coded Decimal(BCD)
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	0101
6	0110	0110
7	0111	0111
8	1000	1000
9	1001	1001
10	1010	0001 0000
11	1011	0001 0001
12	1100	0001 0010
13	1101	0001 0011
14	1110	0001 0100
15	1111	0001 0101

BCD Addition

- **Step 1:** Add the two BCD numbers, using the rules for binary addition.
- **Step 2:** If a 4-bit sum is equal to or less than 9, it is a valid BCD number.
- **Step 3:** If a 4-bit sum is greater than 9, or if a carry out of the 4-bit group is generated, it is an invalid result. Add 6 (0110) to the 4-bit sum in order to skip the six invalid BCD code words and return the code to 8421. If a carry results when 6 is added, simply add the carry to the next 4-bit group.

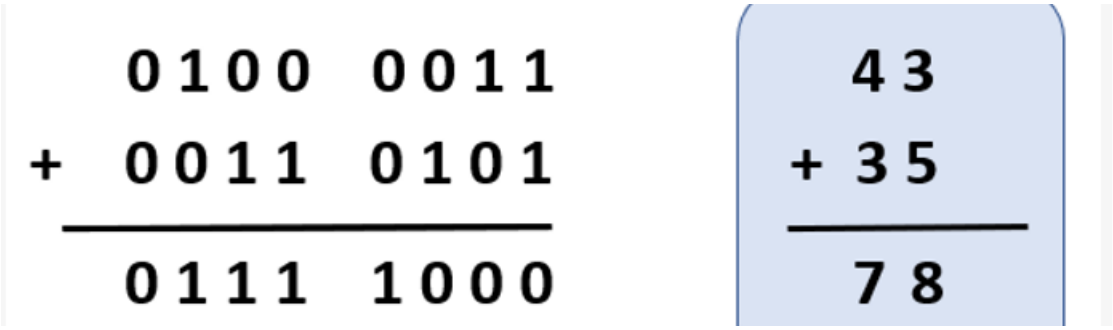
Example 1: Find the sum of the BCD numbers 01000011 + 00110101

Solution:

Decimal number of the given BCD numbers are as below:

$01000011_{BCD} = 43_{10}$

$00110101_{BCD} = 35_{10}$



In the above example, all the 4-bit BCD additions generate valid BCD number, that means less than 9. So, the final correct result is $78_{10} = 01111000_{BCD}$.

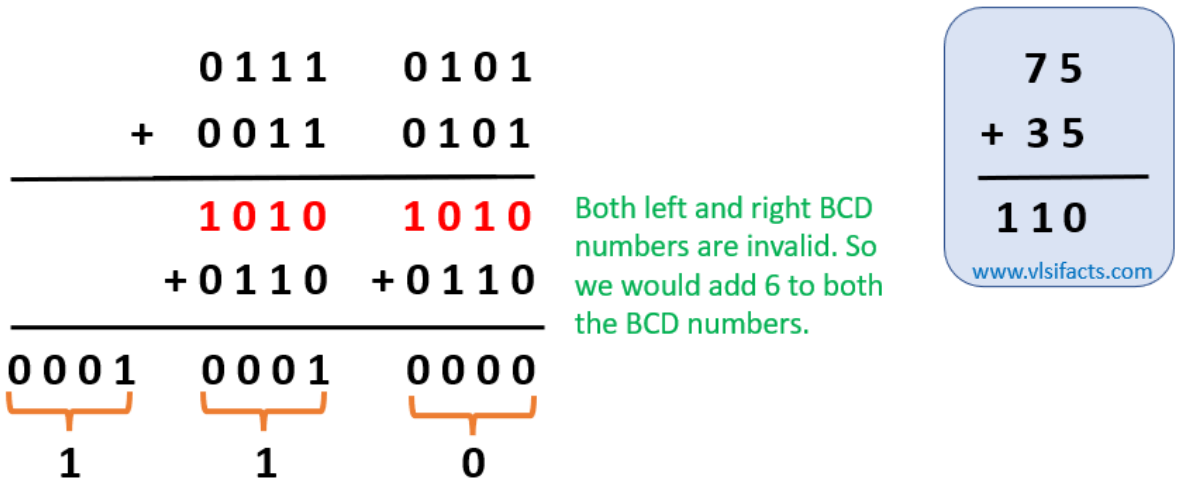
Example 2: Find the sum of the BCD numbers 01110101 + 00110101

Solution:

Decimal number of the given BCD numbers are as below:

$01110101_{BCD} = 75_{10}$

$00110101_{BCD} = 35_{10}$



both the BCD code addition generated result larger than 9, so invalid. To get the correct result, we added 6 (0110_2) to both the invalid sum. Notice that the carry generated from the left group is forwarded to the right group. The final correct result is $110_{10} = 000100010000_{BCD}$.

BCD Subtraction

9's Complement Method

- *Find the 9's complement of the subtrahend*
- *Add it to the minuend using BCD addition*
- *If result is invalid BCD, add 6 (0110)*
- *Shift carry to next bit*

- *If end around carry is generated add carry to the result, otherwise find the 9's complement of the result.*
  Example: Solve BCD Subtraction using 9's Complement Method 987.6 – 678.9
  Solution
    - *Find the 9's complement of the subtrahend* 678.9

	9	9	9.	9
+	6	7	8.	9
	3	2	1.	0

    - *Add it to the minuend using BCD addition*

			1	1							1	1	1					
987.6		1	0	0	1		1	0	0	0		0	1	1	1.	0	1	1	0
321.0	+	0	0	1	1		0	0	1	0		0	0	0	1.	0	0	0	0
		1	1	0	0		1	0	1	0		1	0	0	0.	0	1	1	0
		invalid					invalid					valid					valid		

    - *If result is invalid BCD, add 6 (0110)*

	1					1	1											
	1	1	0	0		1	0	1	0		1	0	0	0.	0	1	1	0
+	0	1	1	0		0	1	1	0									
1	0	0	1	0	1	0	0	0	0		1	0	0	0.	0	1	1	0

    - *Shift carry to next bit*

	0	0	1	0		0	0	0	0		1	0	0	0.	0	1	1	0
+				1														1
	0	0	1	1		0	0	0	0		1	0	0	0.	0	1	1	1
	3					0					8.				7			

    - *no end around carry is generated*
      Answer: 987.6 – 678.9 = 308.7

10's Complement Method

- Find the 10's complement of a negative number
- Add two numbers using BCD addition
- If carry is not generated find the 10s Complement of the result , and assign negative sign
- If end around carry is generated, discard carry

Example: Solve BCD Subtraction using 10's Complement Method 987.6 – 678.9

Solution

- *Find the 10's complement of the subtrahend* 678.9

```
      9   9   9.  9
+     6   7   8.  9
      3   2   1.  0
+                 1
-------------------
      3   2   1.  1
```

- *Add it to the minuend using BCD addition*

```
                1 1                 1 1 1
987.6     1 0 0 1     1 0 0 0     0 1 1 1. 0 1 1 0
321.1  +  0 0 1 1     0 0 1 0     0 0 0 1. 0 0 0 1
-------------------------------------------------
          1 1 0 0     1 0 1 0     1 0 0 0. 0 1 1 1
        invalid     invalid       valid     valid
```

- *If result is invalid BCD, add 6 (0110)*

```
    1               1 1
    1 1 0 0     1 0 1 0     1 0 0 0. 0 1 1 1
+   0 1 1 0     0 1 1 0
-------------------------------------------
1 0 0 1 0 1 0 0 0 0     1 0 0 0. 0 1 1 1
```

- *Discard end around carry and Shift carry to next bit*

```
      0 0 1 0     0 0 0 0     1 0 0 0. 0 1 1 1
+               1           0                 0
-------------------------------------------------
      0 0 1 1     0 0 0 0     1 0 0 0. 0 1 1 1
        3           0           8.          7
```

Answer: 987.6 – 678.9 = 308.7

Example: Solve BCD Subtraction using 10's Complement Method 26.25 – 38.26

Solution

- *Find the 10's complement of the subtrahend* 38.26

```
      9   9.  9   9
+     3   8.  2   6
      6   1.  7   3
+                 1
-------------------
      6   1.  7   4
```

- *Add it to the minuend using BCD addition*

```
                1 1                       1 1           1
26.25     0 0 1 0   0 1 1 0.    0 0 1 0   0 1 0 1
61.74  +  0 1 1 0   0 0 0 1.    0 1 1 1   0 1 0 0
-------------------------------------------------
          1 0 0 0   0 1 1 1.    1 0 0 1   1 0 0 1
            8 valid             7 valid     9 valid     9 valid
```

- *all results is valid BCD*
- *carry is not generated, find the 10s Complement of the result, and assign negative sign*

```
      9   9.  9   9
+     8   7.  9   9
      1   2.  0   0
+                 1
-------------------
      1   2.  0   1
```

Answer: 26.25 – 38.26 = - 12.01

Octal Addition

Example: Solve 456_8 - 123_8
Solution

```
      4   5   6
+     1   2   3
      ?   ?   ?
```

6+3 = 9 = (8x1) + 1
5+2+1 = 8 = (8x1) + 0
4+1+1 = 6

```
      1   1
      4   5   6
+     1   2   3
-------------------
      6   0   1
```

Answer: $456_8 - 123_8 = 601_8$

Octal Subtraction

1. General Method

Example
$743_8 - 564_8$

```
    7  4  3
-   5  6  4
```

3-4 cannot be done, so borrow from preceeding 4,
4<6, so borrow from preceeding 7
So 7 becomes 6

```
    6  8  8          8+3-4= 7
    ~~7~~  4  3          8+4-6= 5
-   5  6  4          6-5= 1
    _____
    1  5  7
```

$743_8 - 564_8 = 157_8$

2. 7's Complement Method

Steps

1. *Find 7's complement of Subtrahend*
2. *Add Minuend to 7's complement of Subtrahend*
3. *If carry is generated, add carry to result. If no carry, find 7"s complement of result and assign negative sign*

Example: Solve $743_8 - 564_8$

1. Find 7's complement of Subtrahend 564

```
   7  7  7
-  5  6  4
_________
```

2 1 3

2. Add Minuend 743 to 7's complement of Subtrahend (213)

	7	4	3
+	2	1	3
1	1	5	6

7+2= 9 = (8 X 1) + 1

3. carry is generated, add carry to result.

	1	5	6
+			1
	1	5	7

Answer: $743_8 - 564_8 = 157_8$

3 8's Complement Method

4. Find 8's complement of Subtrahend
5. Add Minuend to 8's complement of Subtrahend
6. If carry is generated, discard carry. If no carry, find 8"s complement of result and assign negative sign

Example
$743_8 - 564_8$

1. Find 8's complement of Subtrahend 564
   - Find 7's complement of Subtrahend 564

```
  7 7 7
 -5 6 4
 _____
  2 1 3
```

   - Add 1 to 7's complement 213

	2	1	3
+			1
	2	1	4

2. Add Minuend 743 to 8's complement of Subtrahend (214)

	7	4	3
+	2	1	4
1	1	5	7

7+2= 9 = (8 X 1) + 1

3. carry is generated, discard carry

Answer = $743_8 - 564_8 = 157_8$

Hexadecimal Addition

Example: Solve $4A6_{16} – 1B3_{16}$
Solution

	4	A	6
+	1	B	3
	?	?	?

6+3 = 9
A+B = 10+11= 21 = (16X1) + 5
4+1+1 = 6

	1		
	4	A	6
+	1	B	3
	6	5	9

Answer: $4A6_8 – 1B3_{16} = 659_{16}$

Hexadecimal Subtraction

1. General Method

Example
Solve $974B_{16} – 587C_{16}$

	9	7	4	B
-	5	8	7	C
	?	?	?	?

B-C requires borrow. Borrow from MSB 9

	8	15	15	16
	~~9~~	7	4	B
-	5	8	7	C
	?	?	?	?

A=10, B= 11. C=12, D=14, F=15
16+11-12= 15 = F
15+4-7= 12= C

15+7-8= 14= E
8-5 =3

	8	15	15	16
	~~9~~	7	4	B
-	5	8	7	C
	3	E	C	F

Answer: $974B_{16} – 587C_{16} = 3ECF_{16}$

2. <u>15's Complement Method</u>

Steps

a. *Find 15's complement of Subtrahend*
b. *Add Minuend to 15's complement of Subtrahend*
c. *If carry is generated, add carry to result. If no carry, find 15's complement of result and assign negative sign*

Example #1: Solve $974B_{16} – 587C_{16}$
Solution:

a. *Find 15's complement of Subtrahend*

	F	F	F	F
-	5	8	7	C
	?	?	?	?

A=10, B= 11. C=12, D=14, F=15

F-C = 15-12 = 3
F-7 = 15-7 = 8
F-8 = 15-8 = 7
F-5 = 15-5 = 10 = A

	F	F	F	F
-	5	8	7	C
	A	7	8	3

b. *Add Minuend to 15's complement of Subtrahend*

	9	7	4	B
+	A	7	8	3
	?	?	?	?

B+3 = 11+3 = 14 = E
4+8=12= C
7+7=14= E
9+A=9+10=19=(16X1) + 3

	9	7	4	B
+	A	7	8	3

1 3 E C E

c. *carry is generated, add carry to result.*

```
    3  E  C  E
+              1
-------------------
    3  E  C  F
```

Answer: $974B_{16} – 587C_{16} = 3ECF_{16}$

Example #2: Solve $971_{16} –CB1_{16}$
Solution:

a. *Find 15's complement of Subtrahend*

```
    F  F  F
-   C  B  1
-------------
    ?  ?  ?
```

A=10, B= 11. C=12, D=14, F=15

F-1 = 15-1 = 14 = E
F-B=15-11=4
F-C=15-12=3

```
    F  F  F
-   C  B  1
-------------
    3  4  E
```

b. *Add Minuend to 15's complement of Subtrahend*

```
    9  7  1
+   3  4  E
-------------
    ?  ?  ?
```

E+1 = F
7+4= 11= B
9+3= 12= C

```
    9  7  1
+   3  4  E
-------------
    C  B  F
```

c. *no carry, find 15's complement of result and assign negative sign*

```
    F  F  F
-   C  B  F
-------------
    ?  ?  ?
```

F-F = 0
F-B= 15- 11 = 4
F-C = 15-12 = 3

		F	F	F
	-	C	B	F
		3	4	0

Answer: $971_{16} – CB1_{16} = -340_{16}$

3. <u>16's Complement Method</u>

Steps

a. *Find 16's complement of Subtrahend*
b. *Add Minuend to 16's complement of Subtrahend*
c. *If carry is generated, discard carry. If no carry, find 16's complement of result and assign negative sign*

Example
Solve $974B_{16} – 587C_{16}$

a. *Find 16's complement of Subtrahend*

Find 15's complement of Subtrahend

		F	F	F	F
	-	5	8	7	C
		?	?	?	?

A=10, B= 11. C=12, D=14, F=15

F-C = 15-12 = 3
F-7 = 15-7 = 8
F-8 = 15-8 = 7
F-5 = 15-5 = 10 = A

		F	F	F	F
	-	5	8	7	C
		A	7	8	3

Find 16's complement of Subtrahend

		A	7	8	3
	+				1
		A	7	8	4

b. *Add Minuend to 15's complement of Subtrahend*

		9	7	4	B

```
+      A      7      8      4

       ?      ?      ?      ?
```

B+4 = 11+4 = 15 = F
4+8=12= C
7+7=14= E
9+A=9+10=19=(16X1) + 3

```
       9      7      4      B
+      A      7      8      3
-------------------------------
1      3      E      C      F
```

c. *If carry is generated, discard carry.*

```
       3      E      C      E
+                            1
-------------------------------
       3      E      C      F
```

Answer: $974B_{16} – 587C_{16} = 3ECF_{16}$

Example #2: Solve $971_{16} – CB1_{16}$
Solution:

a. *Find 16's complement of Subtrahend*
Find 15's complement of Subtrahend

```
       F      F      F
-      C      B      1
-------------------------
       ?      ?      ?
```

A=10, B= 11. C=12, D=14, F=15

F-1 = 15-1 = 14 = E
F-B=15-11=4
F-C=15-12=3

```
       F      F      F
-      C      B      1
-------------------------
       3      4      E
```

Find 16's complement of Subtrahend

```
       3      4      E
+                    1
-------------------------
       3      4      F
```

a. *Add Minuend to 15's complement of Subtrahend*

```
       9      7      1
+      3      4      F
-------------------------
```

```
      ?   ?   ?
```

F+1 = 16 = (16X1)+0
1+7+4= 12= C
9+3= 12= C

```
        1
    9   7   1
+   3   4   F
-------------
    C   C   0
```

b. no carry, find 16's complement of result and assign negative sign
find 15's complement of result

```
      F   F   F
-     C   C   0
-------------
      ?   ?   ?
```

F-0 = F
F-C= 15- 12 = 3
F-C = 15-12 = 3

```
      F   F   F
-     C   C   0
-------------
      3   3   F
```

find 16's complement of result

```
        1
    3   3   F
+           1
-------------
    3   4   0
```

Answer: $971_{16} – CB1_{16} = -340_{16}$

Binary Codes

When numbers, letters or words are represented by a specific group of symbols, it is said that the number, letter or word is being encoded. The group of symbols is called as a code.

Classification of binary codes

Binary codes are broadly categorized into following categories.

- Weighted Codes
- Non-Weighted Codes

Weighted Codes

Weighted binary codes are those binary codes which obey the positional weight principle. Each position of the number represents a specific weight. Several systems of the codes are used to express the decimal digits 0 through 9 (like 8421, 2421 and 84-2-1). In these codes each decimal digit is represented by a group of four bits.

Weighted codes can be further classified as positively weighted codes (8421, 2421) and negatively weighted codes (84-2-1).

Non-Weighted Codes

In this type of binary codes, the positional weights are not assigned. The examples of non-weighted codes are Excess-3 code and Gray code.

8 4 2 1 code

- The weights of this code are 8, 4, 2 and 1.
- This code has all positive weights. So, it is a positively weighted code.
- This code is also called as natural BCD code.

Example

Let us find the BCD equivalent of the decimal number 786. This number has 3 decimal digits 7, 8 and 6. From the table, we can write the BCD 84218421 codes of 7, 8 and 6 are 0111, 1000 and 0110 respectively.

$$\therefore 786_{10} = 011110000110_{\mathbf{BCD}}$$

There are 12 bits in BCD representation, since each BCD code of decimal digit has 4 bits.

2 4 2 1 code

- The weights of this code are 2, 4, 2 and 1.
- This code has all positive weights. So, it is a positively weighted code.
- It is an unnatural BCD code. Sum of weights of unnatural BCD codes is equal to 9.
- It is a self-complementing code. Self-complementing codes provide the 9's complement of a decimal number, just by interchanging 1's and 0's in its equivalent 2421 representation.

Example

Let us find the 2421 equivalent of the decimal number 786. This number has 3 decimal digits 7, 8 and 6. From the table, we can write the 2421 codes of 7, 8 and 6 are 1101, 1110 and 1100 respectively.

Therefore, the 2421 equivalent of the decimal number 786 is **110111101100**.

8 4 -2 -1 code

- The weights of this code are 8, 4, -2 and -1.
- This code has negative weights along with positive weights. So, it is a negatively weighted code.

- It is an unnatural BCD code.
- It is a self-complementing code.

Example

Let us find the 8 4-2-1 equivalent of the decimal number 786. This number has 3 decimal digits 7, 8 and 6. From the table, we can write the 8 4 -2 -1 codes of 7, 8 and 6 are 1001, 1000 and 1010 respectively.

Therefore, the 8 4 -2 -1 equivalent of the decimal number 786 is 100110001010.

Excess 3 code

- This code doesn't have any weights. So, it is an un-weighted code.
- We will get the Excess 3 code of a decimal number by adding three 0011 to the binary equivalent of that decimal number. Hence, it is called as Excess 3 code.
- It is a self-complementing code.

Example

Let us find the Excess 3 equivalent of the decimal number 786. This number has 3 decimal digits 7, 8 and 6. From the table, we can write the Excess 3 codes of 7, 8 and 6 are 1010, 1011 and 1001 respectively.

Therefore, the Excess 3 equivalent of the decimal number 786 is 101010111001

Example 1: Decimal number 31

1. We find the BCD code of each digit of the decimal number.

Digit	BCD
3	0011
1	0001

2) Then, we add 0011 in both of the BCD code.

Decimal	BCD	Excess-3
3	0011+0011	0110
1	0001+0011	0100

3. So, the excess-3 code of the decimal number 31 is 0110 0100

Example 2: Decimal number 81.61

- We find the BCD code of each digit of the decimal number.

Digit	BCD
8	1000

1 0001

6 0110

1 0001

- Then, we add 0011 in both of the BCD code.

Decimal	BCD	Excess-3
8	1000+0011	1011
1	0001+0011	0100
6	0110+0011	1001

- So, the excess-3 code of the decimal number 81.61 is 1011 0100.1001 0100

Self-complementary property

A self-complementary binary code is a code which is always complimented in itself. By replacing the bit 0 to 1 and 1 to 0 of a number, we find the 1's complement of the number. The sum of the 1'st complement and the binary number of a decimal is equal to the binary number of decimal 9.

Binary Codes for Decimal digits

The following table shows the various binary codes for decimal digits 0 to 9.

Decimal Digit	8421 Code	2421 Code	84-2-1 Code	Excess 3 Code
0	0000	0000	0000	0011
1	0001	0001	0111	0100
2	0010	0010	0110	0101
3	0011	0011	0101	0110
4	0100	0100	0100	0111
5	0101	1011	1011	1000
6	0110	1100	1010	1001
7	0111	1101	1001	1010
8	1000	1110	1000	1011
9	1001	1111	1111	1100

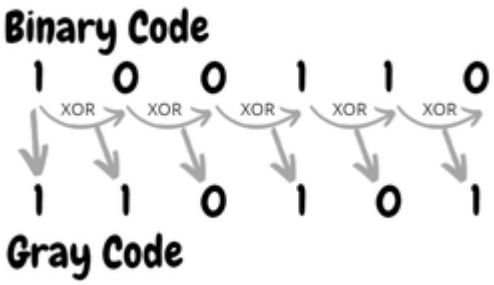
Gray Codes

The **Gray Code** is a sequence of binary number systems, which is also known as **reflected binary code**. The reason for calling this code as reflected binary code is the first N/2 values compared with those of the last N/2 values in reverse order. In this code, two consecutive values are differed by one bit of binary digits. Gray codes are used in the general sequence of hardware-generated binary numbers. These numbers cause ambiguities or errors when the transition from one number to its successive is done. This code simply solves this problem by changing only one bit when the transition is between numbers is done.

The gray code is a very light weighted code because it doesn't depend on the value of the digit specified by the position. This code is also called a cyclic variable code as the transition of one value to its successive value carries a change of one bit only.

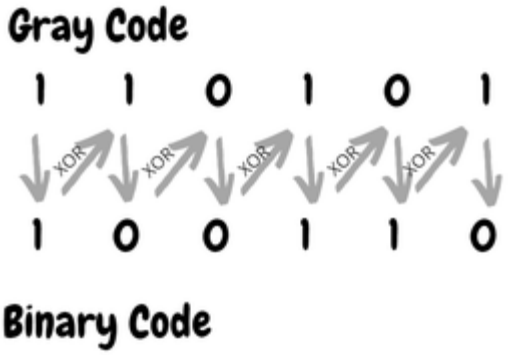
Binary to Gray conversion:

1. The Most Significant Bit (MSB) of the gray code is always equal to the MSB of the given binary code.
2. Other bits of the output gray code can be obtained by XORing binary code bit at that index and previous index.



Gray to binary conversion:

1. The Most Significant Bit (MSB) of the binary code is always equal to the MSB of the given gray code.
2. Other bits of the output binary code can be obtained by XORing gray code bit at that index and previous binary code bit.



Decimal Number	Binary Number	Gray Code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Error Detection and Correction Code

Error detection and correction code plays an important role in the transmission of data from one source to another. The noise also gets added into the data when it transmits from one system to another, which causes errors in the received binary data at other systems. The bits of the data may change (either 0 to 1 or 1 to 0) during transmission.

It is impossible to avoid the interference of noise, but it is possible to get back the original data. For this purpose, we first need to detect either an error **z** is present or not using error detection codes. If the error is present in the code, then we will correct it with the help of error correction codes

Error detection code

The error detection codes are the code used for detecting the error in the received data **bitstream**. In these codes, some bits are included appended to the original bitstream.

Error detecting codes encode the message before sending it over the noisy channels. The encoding scheme is performed in such a way that the decoder at the receiving can find the errors easily in the receiving data with a higher chance of success.

Parity Code

In parity code, we add one parity bit either to the right of the LSB or left to the MSB to the original bitstream. On the basis of the type of parity being chosen, two types of parity codes are possible, i.e., even parity code and odd parity code.

Error correction code

Error correction codes are generated by using the specific algorithm used for removing and detecting errors from the message transmitted over the noisy channels. The error-correcting codes find the correct number of corrupted bits and their positions in the message. There are two types of ECCs(Error Correction Codes), which are as follows.

Block codes

in block codes, in fixed-size blocks of bits, the message is contained. In this, the redundant bits are added for correcting and detecting errors.

Hamming Code

Hamming code is an example of a block code. The two simultaneous bit errors are detected, and single-bit errors are corrected by this code. In the hamming coding mechanism, the sender encodes the message by adding the unessential bits in the data. These bits are added to the specific position in the message because they are the extra bits for correction.