Subject Name: Digital Electronics
Subject Code: BCA10100
Unit No. :-1
Topic Name :- Number Systems and Binary Arithmetic

Dr. Sumit Gupta
Department of Computer Science & Applications

Course Structure

Course Code	BCA10100				
Course Category	Program Foundation				
Course Title	Introduction to Digital Electronics				
Teaching Scheme	Lectures	Tutorials	Laboratory / Practical	Project	Total
Weekly load hours	3	-	-	-	3
Credits	3	-	-	-	3
Assessment Schema Code	TT1				

Assessment Structure

Type of Course	Assessment Scheme Code	Description L-T-P-J-C	CCA1	MT	CCA2	LCA1	LCA2	LCA3	TE	Total
Theory Courses	TT1	All Theory (L, T) Only courses **with** TE exams	15	30	15	-	-	-	40	100

Course Outcomes

Course Outcomes:

After the completion of this course students will be able to:

1. Perform conversions and binary arithmetic operations, including BCD and Gray code, using various number systems.
2. Simplify Boolean expressions and design logic circuits using Boolean algebra and logic gates.
3. Design and implement basic combinational circuits such as adders, multiplexers, and encoders.
4. Design sequential circuits, including flip-flops, shift registers, and counters, using excitation tables.

Course Contents

Unit 1: Number Systems and Binary Arithmetic [8]

Introduction to Decimal, Binary, Octal and Hexadecimal number systems and their inter-conversions, Signed and fractional binary number representations, Binary Coded Decimal, Gray Codes, Gray to Binary and Binary to Gray conversion, Alphanumeric representation in ASCII codes. Rules of binary addition and subtraction, and subtraction using 1's and 2's complements.

Resources

Learning Resources:

Textbooks:

1. Digital Design, M. Morris Mano, 3rd Edition 2022, Publisher: Pearson.
2. Digital systems : principles and applications / Ronald J.Tocci, Neal S. Widmer, Gregory L. Moss.—10th ed. 2007, Pearson Education, Inc., Upper Saddle River, New Jersey 07458.
3. Digital Electronics, G.K. Kharate, Oxford; Reprint edition (11 March 2010).
4. Digital Circuits and Design, S. Salivahanan, S. Arivazhagan, 5th Edition, Oxford University Press; Fifth edition (1 March 2018); Oxford University Press.

Reference Books:

1. Digital Fundamentals: Floyd T.M., Jain R.P., Pearson Education, 5th Edition.
2. Digital Electronics: Jain R.P., Tata McGraw Hill, 3rd Edition.
3. Digital Principles and Applications: Malvino Leach, Tata McGraw-Hill, 2nd Edition.

Content

- ✓ Introduction to Decimal, Binary, Octal and Hexadecimal number systems
- ✓ Number system inter-conversions,
- ✓ Signed and fractional binary number representations,
- ✓ Binary Coded Decimal, Gray Codes,
- ✓ Gray to Binary and Binary to Gray conversion,
- ✓ Alphanumeric representation in ASCII codes.
- ✓ Rules of binary addition and subtraction,
- ✓ Subtraction using 1's and 2's complements.

Digital System

- ✓ A Digital system is an interconnection of digital modules.
- ✓ It is a system that manipulates discrete elements of information that is represented internally in the binary form.
- ✓ Like
  - ✓ Automated industrial machinery
  - ✓ Pocket calculators
  - ✓ Microprocessors
  - ✓ Digital computers
  - ✓ Digital watches
  - ✓ TV games
  - ✓ Signal processing and so on.

Digital System

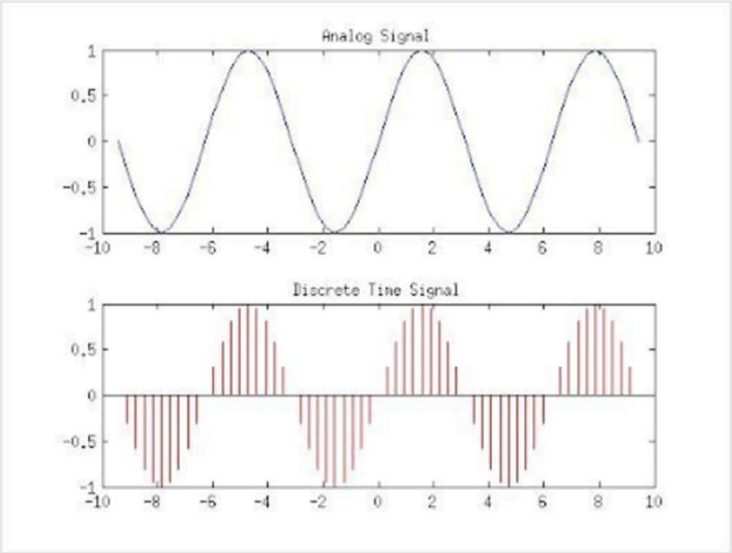
Characteristics of Digital systems

- ✓ Digital systems manipulate discrete elements of information.
- ✓ Discrete elements are nothing but the digits such as 10 decimal digits or 26 letters of alphabets and so on.
- ✓ Digital systems use physical quantities called signals to represent discrete elements.
- ✓ In digital systems, the signals have two discrete values and are therefore said to be binary.
- ✓ A signal in digital system represents one binary digit called a bit. The bit has a value either 0 or 1.

Digital System

Analog Vs Discrete Vs Digital systems

The first step of digitization is to sample the analog signal. Output of this process is a discrete time signal. At this point, there is no constraint on amplitude of the output signal. Figure below shows both the signals.



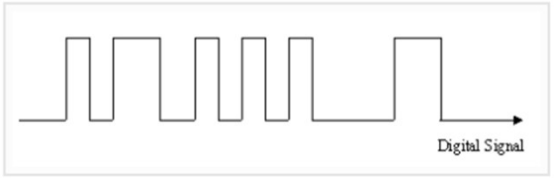
Digital System

Analog Vs Discrete Vs Digital systems

The next step of the digitization process is quantization. It involves approximating the sampled analog values using a finite set of values. After this step, the output signal is discrete in both time and amplitude. Amplitude of the signal takes 1 of the M possible values.

The third step of digitization is to encode these quantized values into bits. Each amplitude level can be represented as a binary sequence. The output of this process is a digital signal that is a physical signal. It is continuous in time and amplitude of this signal is either 1 or 0.

Digital signal can also refer to a signal that switches between discrete number of voltage levels.



Number System

- ✓ Number system is a basis for counting varies items.
- ✓ Modern computers communicate and operate with binary numbers which use only the digits 0 &1.
- ✓ Basic number system used by humans is Decimal number system.

For Ex: Let us consider decimal number 18. This number is represented in binary as 10010.

We observe that binary number system take more digits to represent the decimal number. For large numbers, we have to deal with very large binary strings. So this fact gave rise to three new number systems.

i) Octal number systems
ii) Hexadecimal number system
iii) Binary Coded Decimal number(BCD) system

Number System

To define any number system, we have to specify

- ✓ Base of the number system such as 2,8,10 or 16
- ✓ The base decides the total number of digits available in that number system
- ✓ First digit in the number system is always zero and last digit in the number system is always base-1.

It can have different base values like: binary (base-2), octal (base-8), decimal (base 10) and hexadecimal (base 16), here the base number represents the number of digits used in that numbering system. As an example, in decimal numbering system, the digits used are: 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9. Therefore the digits for binary are: 0 and 1, the digits for octal are: 0, 1, 2, 3, 4, 5, 6 and 7. For the hexadecimal numbering system, base 16, the digits are: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Decimal Number System

- ✓ The **base** or radix of Decimal number system is **10**.
- ✓ So, the numbers ranging from 0 to 9 are used in this number system.
- ✓ The part of the number that lies to the left of the **decimal point** is known as integer part.
- ✓ Similarly, the part of the number that lies to the right of the decimal point is known as fractional part.
- ✓ In this number system, the successive positions to the left of the decimal point having weights of 10^0, 10^1, 10^2, 10^3 and so on.
- ✓ Similarly, the successive positions to the right of the decimal point having weights of 10^{-1}, 10^{-2}, 10^{-3} and so on. That means, each position has specific weight, which is **power of base 10**

Example

Consider the **decimal number 1358.246**. Integer part of this number is 1358 and fractional part of this number is 0.246.

The digits 8, 5, 3 and 1 have weights of 10^0, 10^1, 10^2 and 10^3 respectively. Similarly, the digits 2, 4 and 6 have weights of 10^{-1}, 10^{-2} and 10^{-3} respectively.

Mathematically, we can write it as

$1358.246 = (1 \times 10^3) + (3 \times 10^2) + (5 \times 10^1) + (8 \times 10^0) + (2 \times 10^{-1}) + (4 \times 10^{-2}) + (6 \times 10^{-3})$

After simplifying the right hand side terms, we will get the decimal number, which is on left hand side.

Binary Number System

- ✓ Numbers that contain only two digit 0 and 1 are called Binary Numbers.
- ✓ Each 0 or 1 is called a Bit, from binary digit.
- ✓ A binary number of 4 bits is called a Nibble.
- ✓ A binary number of 8 bits is called a Byte. A binary number of 16 bits is called a Word on some systems, on others a 32-bit number is called a Word while a 16-bit number is called a Halfword.
- ✓ Using 2 bit 0 and 1 to form
  - ✓ a binary number of 1 bit, numbers are 0 and 1
  - ✓ a binary number of 2 bit, numbers are 00, 01, 10, 11
  - ✓ a binary number of 3 bit, such numbers are 000, 001, 010, 011, 100, 101, 110, 111
  - ✓ a binary number of 4 bit, such numbers are 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100,1101,1110,1111

  Therefore , using n bits there are 2^n binary numbers of n bits.

Binary Number System

- ✓ The part of the number, which lies to the left of the **binary point** is known as integer part.
- ✓ Similarly, the part of the number, which lies to the right of the binary point is known as fractional part.
- ✓ In this number system, the successive positions to the left of the binary point having weights of 2^0, 2^1, 2^2, 2^3 and so on.
- ✓ Similarly, the successive positions to the right of the binary point having weights of 2^{-1}, 2^{-2}, 2^{-3} and so on. That means, each position has specific weight, which is **power of base 2**.

Example
Consider the **binary number 1101.011**. Integer part of this number is 1101 and fractional part of this number is 0.011.
The digits 1, 0, 1 and 1 of integer part have weights of 2^0, 2^1, 2^2, 2^3 respectively. Similarly, the digits 0, 1 and 1 of fractional part have weights of 2^{-1}, 2^{-2}, 2^{-3} respectively.

Mathematically, we can write it as

$1101.011 = (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) + (0 \times 2^{-1}) + (1 \times 2^{-2}) + (1 \times 2^{-3})$
After simplifying the right hand side terms, we will get a decimal number, which is an equivalent of binary number on left hand side.

Octal Number System

- ✓ The **base** or radix of octal number system is **8**.
- ✓ So, the 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.
- ✓ Similarly, the part of the number that lies to the right of the octal point is known as fractional part.
- ✓ In this number system, the successive positions to the left of the octal point having weights of 8^0, 8^1, 8^2, 8^3 and so on.
- ✓ Similarly, the successive positions to the right of the octal point having weights of 8^{-1}, 8^{-2}, 8^{-3} and so on. That means, each position has specific weight, which is **power of base 8**.

Example

- ✓ Consider the **octal number 1457.236**. Integer part of this number is 1457 and fractional part of this number is 0.236.
- ✓ The digits 7, 5, 4 and 1 have weights of 8^0, 8^1, 8^2 and 8^3 respectively. Similarly, the digits 2, 3 and 6 have weights of 8^{-1}, 8^{-2}, 8^{-3} respectively.

Mathematically, we can write it as

$1457.236 = (1 \times 8^3) + (4 \times 8^2) + (5 \times 8^1) + (7 \times 8^0) + (2 \times 8^{-1}) + (3 \times 8^{-2}) + (6 \times 8^{-3})$

After simplifying the right hand side terms, we will get a decimal number, which is an equivalent of octal number on left hand side.

Hexadecimal Number System

- ✓ The **base** or radix of Hexa-decimal number system is **16**.
- ✓ So, the 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.
- ✓ Similarly, the part of the number, which lies to the right of the Hexa-decimal point is known as fractional part.
- ✓ In this number system, the successive positions to the left of the Hexa-decimal point having weights of 16^0, 16^1, 16^2, 16^3 and so on.
- ✓ Similarly, the successive positions to the right of the Hexa-decimal point having weights of 16^{-1}, 16^{-2}, 16^{-3} and so on. That means, each position has specific weight, which is **power of base 16**.

Example

- ✓ Consider the **Hexa-decimal number 1A05.2C4**. Integer part of this number is 1A05 and fractional part of this number is 0.2C4.
- ✓ The digits 5, 0, A and 1 have weights of 16^0, 16^1, 16^2 and 16^3 respectively. Similarly, the digits 2, C and 4 have weights of 16^{-1}, 16^{-2} and 16^{-3} respectively.

Mathematically, we can write it as

$1A05.2C4 = (1 \times 16^3) + (10 \times 16^2) + (0 \times 16^1) + (5 \times 16^0) + (2 \times 16^{-1}) + (12 \times 16^{-2}) + (4 \times 16^{-3})$

After simplifying the right hand side terms, we will get a decimal number, which is an equivalent of Hexa-decimal number on left hand side.

Decimal Number to other Bases Conversion

- ✓ If the decimal number contains both integer part and fractional part, then convert both the parts of decimal number into other base individually.
- ✓ Follow these steps for converting the decimal number into its equivalent number of any base ‘r’.
  - ✓ Do **division** of integer part of decimal number and **successive quotients** with base ‘r’ and note down the remainders till the quotient is zero. Consider the remainders in reverse order to get the integer part of equivalent number of base ‘r’ i.e., **first and last remainders denote the least significant digit and most significant digit respectively.**
  - ✓ Do **multiplication** of fractional part of decimal number and **successive fractions** with base ‘r’ and note down the carry till the result is zero or the desired number of equivalent digits is obtained. Consider the normal sequence of carry in order to get the fractional part of equivalent number of base ‘r’.

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 2.

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
1/2	0	1 MSB

$$\Rightarrow 58_{10} = 111010_{2}$$

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

Decimal to Binary Conversion

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
-	0.0	-

$$\Rightarrow .25_{10} = .01_{2}$$

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.

Decimal to Binary Conversion

Exercise – Conversion of the following number in Binary Number-

1. 25_{10}
2. 48_{10}
3. 50.35_{10}
4. 147.98_{10}
5. 98_{10}

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**.

Decimal to Octal Conversion

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.

Decimal to Octal Conversion

Exercise – Conversion of the following number in Octal Number-

1. 25_{10}
2. 48_{10}
3. 50.35_{10}
4. 147.98_{10}
5. 98_{10}

Decimal to Hexadecimal Conversion

- ✓ The following two types of operations take place, while converting decimal number into its equivalent hexadecimal 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 fractional 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 octal number is **3A**.

Decimal to Hexadecimal Conversion

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 Hexadecimal number is **.4**

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

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

Decimal to Hexadecimal Conversion

Exercise – Conversion of the following number in hexadecimal Number-

1. 25_{10}
2. 48_{10}
3. 50.35_{10}
4. 147.98_{10}
5. 98_{10}

Binary Number to Other Bases Conversion

The process of converting a number from binary to decimal is different to the process of converting a binary number to other bases.

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 -1

Consider the binary number 1101.11.

Mathematically, we can write it as

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

$1101.11 = 8 + 4 + 0 + 1 + 0.5 + 0.25 = 13.75$

Example -2

Consider the binary number 111010.01.

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.

Binary to Octal Conversion

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.

101 110.011 010

Step 2 − Write the octal digits corresponding to each group of 3 bits.

101110.011010 = 56.32

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

Binary to Hexadecimal 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.

Binary to Hexadecimal Conversion

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.

0010 1110.0110 1000

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

00101110.01101000 = 2E.68

Therefore, the **Hexa-decimal equivalent** of binary number 101110.0101 is 2E.68

Octal Number to Other Bases Conversion

The process of converting a number from octal to decimal is different to the process of converting an octal number to other bases.

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.

Octal Number to Other Bases Conversion

The process of converting a number from octal to decimal is different to the process of converting an octal number to other bases.

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.
Mathematically, we can write it as

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.

Octal Number to Other Bases Conversion

The process of converting a number from octal to decimal is different to the process of converting an octal number to other bases.

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*.

Hexa-Decimal to Other Bases Conversion

The process of converting a number from Hexa-decimal to decimal is different to the process of converting Hexa-decimal number into other bases.

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.

Hexa-Decimal to Other Bases Conversion

The process of converting a number from Hexa-decimal to decimal is different to the process of converting Hexa-decimal number into other bases.

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.

$$65.4C_{6} = 01100101.01001100_{2}$$

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.

Hexa-Decimal to Other Bases Conversion

The process of converting a number from Hexa-decimal to decimal is different to the process of converting Hexa-decimal number into other bases.

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 $65.4C$ is 145.23.

Binary Numbers Representation

We can make the binary numbers into the following two groups −

- ✓ Unsigned numbers
- ✓ Signed numbers.

Unsigned Numbers:

Unsigned numbers contain only magnitude of the number. They don't have any sign. That means all unsigned binary numbers are positive. As in decimal number system, the placing of positive sign in front of the number is optional for representing positive numbers. Therefore, all positive numbers including zero can be treated as unsigned numbers.

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.

Binary Numbers Representation

Signed Numbers:

Signed numbers contain both sign and magnitude of the number. Generally, the sign is placed in front of number. So, we have to consider the positive sign for positive numbers and negative sign for negative numbers. Therefore, all numbers can be treated as signed numbers if the corresponding sign is assigned in front of the number.

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.

Binary Numbers Representation

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.

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.

Binary Numbers Representation

Complements

Complements are used in digital computers for simplifying the subtraction operation and for logical manipulation. There are two types of complements for each base ***r*** system:

- ✓ the ***r's*** complement
- ✓ the ***(r-1)'s*** complement

When the value of the base ***r*** is substituted in the name, the two types are referred to as the 2's and 1's complement for binary numbers and the 10's and 9's complement for decimal numbers.

(r-1)'s Complement

Given a number ***N*** in base ***r*** having ***n*** digits, the ***(r-1)'s*** complement of ***N*** is defined as $(r^n - 1) - N$. for example- 9's complement of $(54)_{10} = (10^2 - 1) - 54 = (100 - 1) - 54 = 99 - 54 = (45)_{10}$ We can say that for ***(r-1)'s*** complement, we have to subtract each digit of number from (r-1).

(r 's) Complement

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

Binary Numbers Representation

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

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}.

Binary Numbers Representation

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 i.e, if you perform two times 2's complement of a binary number including sign bit, then you will get the original signed binary number. **The Range of 2's complement is [(-) 2^{n-1} to (+) 2^{n-1} -1]**.

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.

Binary Arithmetic

Binary arithmetic is essential part of all the digital computers and many other digital system.

Binary Addition :
It is a key for binary subtraction, multiplication, division. There are four rules of binary addition.

In fourth case, a binary addition is creating a sum of (1 + 1 = 10) i.e. 0 is written in the given column and a carry of 1 over to the next column.

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

Example – Addition

0011010 + 001100 = 00100110

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

Binary Arithmetic

Binary arithmetic is essential part of all the digital computers and many other digital system.

Binary Subtraction:
Subtraction and Borrow, these two words will be used very frequently for the 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	1	1

Example − Subtraction

0011010 - 001100 = 00001110

1 1	borrow
0 0 ~~1 1~~ 0 1 0	$= 26_{10}$
- 0 0 0 1 1 0 0	$= 12_{10}$
0 0 0 1 1 1 0	$= 14_{10}$

Binary Arithmetic

Binary arithmetic is essential part of all the digital computers and many other digital system.

Binary Multiplication:
Binary multiplication is similar to decimal multiplication. It is simpler than decimal multiplication because only 0s and 1s are involved. There are four rules of 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 − Multiplication

Example:

0011010 x 001100 = 100111000

$$
\begin{array}{rl}
 & 0011010 = 26_{10} \\
\times & 0001100 = 12_{10} \\
\hline
 & 0000000 \\
 & 0000000 \\
 & 0011010 \\
 & 0011010 \\
\hline
 & 0100111000 = 312_{10}
\end{array}
$$

Binary Arithmetic

Binary arithmetic is essential part of all the digital computers and many other digital system.

Binary Division:
Binary division is similar to decimal division. It is called as the long division procedure.

Example − Division

101010 / 000110 = 000111

```
            1 1 1       = 7₁₀
000110 ) 1 0 1 0 1 0    = 42₁₀
         - 1 1 0        = 6₁₀
         -------
           1 0 0 1
          - 1 1 0
          -------
              1 1 0
            - 1 1 0
            -------
                  0
```

Addition of Signed Binary Numbers using 1's complement

Consider the two signed binary numbers A & B, which are represented in **1's complement form**. We can perform the addition of these two numbers, which is similar to the addition of two unsigned binary numbers. But, if the resultant sum contains carry out from sign bit, then it must be added to result and known as **End Around Carry** in order to get the correct value.

Case-1 If the carry is generated, carry is added to result which is found in step 2 and if sign bit is 0 then your result is positive and that is your final result.

Case-2 If sign bit is 1 and no carry generated, your result is Negative and you have to take 1's complement form.

Case-3 If sign bit is 1 and carry generated, your result is Negative and you have to take add that carry in the result found in step 2 then take 1's complement form.

Example 1- Let us perform the addition of two decimal numbers +7 and +4 using 2's complement method. The 2's complement representations of +7 and +4 with 5 bits each are shown below.

Addition of Signed Binary Numbers using 2's complement

Consider the two signed binary numbers A & B, which are represented in **2's complement form**. We can perform the addition of these two numbers, which is similar to the addition of two unsigned binary numbers. **But, if the resultant sum contains carry out from sign bit, then discard in order to get the correct value.**

If resultant sum is positive, you can find the magnitude of it directly. But, if the resultant sum is negative, then take 2's complement of it in order to get the magnitude.

Example 1- Let us perform the addition of two decimal numbers +7 and +4 using 2's complement method. The 2's complement representations of +7 and +4 with 5 bits each are shown below.

$$+7_{10} = 00111_2$$

$$+4_{10} = 00100_2$$

The addition of these two numbers is

$$+7_{10} + +4_{10} = 00111_2 + 00100_2$$

$$\Rightarrow +7_{10} + +4_{10} = 01011_2.$$

The resultant sum contains 5 bits. So, there is no carry out from sign bit. The sign bit '0' indicates that the resultant sum is positive. So, the magnitude of sum is 11 in decimal number system. Therefore, addition of two positive numbers will give another positive number.

Addition of Two Signed Binary Numbers

Example 2- Let us perform the addition of two decimal numbers -7 and -4 using 2's complement method. The 2's complement representation of -7 and -4 with 5 bits each are shown below.

$$-7_{10} = 11001_2$$

$$-4_{10} = 11100_2$$

The addition of these two numbers is

$$-7_{10} + -4_{10} = 11001_2 + 11100_2$$

$$\Rightarrow -7_{10} + -4_{10} = 110101_2.$$

The resultant sum contains 6 bits. In this case, carry is obtained from sign bit. So, we can remove it

Resultant sum after removing carry is $-7_{10} + -4_{10} = 10101_2$.

The sign bit '1' indicates that the resultant sum is **negative**. So, by taking 2's complement of it we will get the magnitude of resultant sum as 11 in decimal number system. Therefore, addition of two negative numbers will give another negative number.

Subtraction of Two Signed Binary Numbers

Consider the two signed binary numbers A & B, which are represented in 2's complement form. We know that 2's complement of positive number gives a negative number. So, whenever we have to subtract a number B from number A, then take 2's complement of B and add it to A. So, mathematically we can write it as

$$A - B = A + \ 2'scomplementofB$$

Similarly, if we have to subtract the number A from number B, then take 2's complement of A and add it to B. So, **mathematically** we can write it as

$$B - A = B + \ 2'scomplementofA$$

So, the subtraction of two signed binary numbers is similar to the addition of two signed binary numbers. But, we have to take 2's complement of the number, which is supposed to be subtracted. This is the **advantage** of 2's complement technique. Follow, the same rules of addition of two signed binary numbers.

Subtraction of Two Signed Binary Numbers

Example 3: Let us perform the subtraction of two decimal numbers +7 and +4 using 2's complement method.

The subtraction of these two numbers is

$$+7_{10} - +4_{10} = +7_{10} + -4_{10}.$$

The **2's complement** representation of +7 and -4 with 5 bits each are shown below.

$$+7_{10} = 00111_2$$

$$+4_{10} = 11100_2$$

$$\Rightarrow +7_{10} + +4_{10} = 00111_2 + 11100_2 = 00011_2$$

Here, the carry obtained from sign bit. So, we can remove it. The resultant sum after removing carry is

$$+7_{10} + +4_{10} = 00011_2$$

The sign bit '0' indicates that the resultant sum is **positive**. So, the magnitude of it is 3 in decimal number system. Therefore, subtraction of two decimal numbers +7 and +4 is +3.

Subtraction of Two Signed Binary Numbers

Example 4

Let us perform the **subtraction of** two decimal numbers **+4** and **+7** using 2's complement method.

The subtraction of these two numbers is

$$+4_{10} - +7_{10} = +4_{10} + -7_{10}.$$

The **2's complement** representation of +4 and -7 with 5 bits each are shown below.

$$+4_{10} = 00100_2$$

$$-7_{10} = 11001_2$$

$$\Rightarrow +4_{10} + -7_{10} = 00100_2 + 11001_2 = 11101_2$$

Here, carry is not obtained from sign bit. The sign bit '1' indicates that the resultant sum is **negative**. So, by taking 2's complement of it we will get the magnitude of resultant sum as 3 in decimal number system. Therefore, subtraction of two decimal numbers +4 and +7 is -3.

Numerical using 1's Complement

1. 2-9 = -7
2. 9-8 = 1
3. 4+6 = 10
4. -2-6 = -8
5. -6-6 = -12
6. 20-15=5
7. 21+8=30
8. -35+15= -20

Numerical using 2's Complement

1. 2-9 = -7
2. 9-8 = 1
3. 4+6 = 10
4. -2-6 = -8
5. -6-6 = -12
6. 20-15=5
7. 21+8=30
8. -25+15=-10

Digital Circuits - Codes

In the coding, when numbers or letters are represented by a specific group of symbols, it is said to be that number or letter is being encoded. The group of symbols is called as code. The digital data is represented, stored and transmitted as group of bits. This group of bits is also called as binary code.

Advantages of Binary Code : Following is the list of advantages that binary code offers.

- ✓ Binary codes are suitable for the computer applications.
- ✓ Binary codes are suitable for the digital communications.
- ✓ Binary codes make the analysis and designing of digital circuits if we use the binary codes.
- ✓ Since only 0 & 1 are being used, implementation becomes easy.

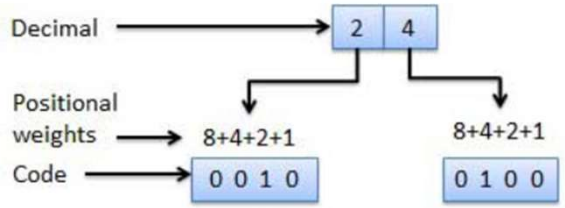
Classification of binary codes : The codes are broadly categorized into the following six categories-

- ✓ Weighted Codes
- ✓ Non-Weighted Codes
- ✓ Binary Coded Decimal Code
- ✓ Alphanumeric Codes
- ✓ Error Detecting Codes
- ✓ Error Correcting Codes

Digital Circuits - 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. In these codes each decimal digit is represented by a group of four bits.



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.

Digital Circuits - Codes

Binary Coded Decimal (BCD) code :

In this code, each decimal digit is represented by a 4-bit binary number. BCD is a way to express each of the decimal digits with a binary code. With four binary bits, we can represent sixteen numbers (0000 to 1111). But in BCD code, only first ten of these are used (0000 to 1001). The remaining six code combinations i.e. 1010 to 1111 are invalid in BCD.

Decimal	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Advantages of BCD Codes :

- ✓ It is very similar to decimal system.
- ✓ We need to remember binary equivalent of decimal numbers 0 to 9 only.

Disadvantages of BCD Codes :

- ✓ The addition and subtraction of BCD have different rules.
- ✓ The BCD arithmetic is little more complicated.
- ✓ BCD needs more number of bits than binary to represent the decimal number. So BCD is less efficient than binary.

Digital Circuits - Codes

Gray Code :

It is the non-weighted code and it is not arithmetic codes. That means there are no specific weights assigned to the bit position. It has a very special feature that, only one bit will change each time. As only one bit changes at a time, the gray code is called as a unit distance code. The gray code is a cyclic code. Gray code cannot be used for arithmetic operation.

Decimal	BCD	Gray
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 1
3	0 0 1 1	0 0 1 0
4	0 1 0 0	0 1 1 0
5	0 1 0 1	0 1 1 1
6	0 1 1 0	0 1 0 1
7	0 1 1 1	0 1 0 0
8	1 0 0 0	1 1 0 0
9	1 0 0 1	1 1 0 1

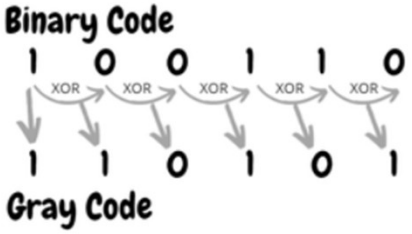
Application of Gray code :

- ✓ Gray code is popularly used in the shaft position encoders.
- ✓ A shaft position encoder produces a code word which represents the angular position of the shaft.

Code Conversion

Binary code to Gray Code Conversion : Follow these steps for converting a binary code into its equivalent Gray code.

- ✓ Keep the MSB of given binary number as it is in Gray code.
- ✓ Compare the successive two bits starting from MSB. If the 2 bits are same, then the output is zero. Otherwise, output is one. (XOR Gate)
- ✓ Repeat the above step till the LSB of Gray code is obtained.



Example:

Given, binary code is 100110.

Step 1 − Keep the MSB of the binary code as it is like 1 in this case.

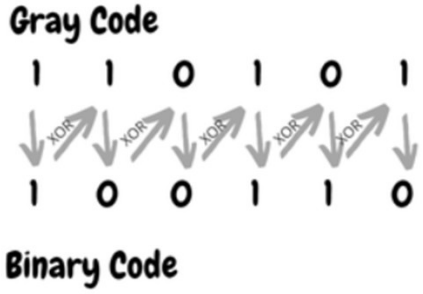
Step 2 − By comparing successive two bits of binary code, we will get the gray code as 110101.

Exercise: Find the Gray code of the following numbers-
(i) 10101 (ii) 1000

Code Conversion

Gray Code to Binary code Conversion : Follow these steps for converting a Gray code into its equivalent binary code.

- ✓ Keep the MSB of given Gray number as it is in binary code.
- ✓ Compare the successive two bits diagonally starting from MSB. If the 2 bits are same, then the output is zero. Otherwise, output is one. (XOR Gate)
- ✓ Repeat the above step till the LSB of Binay code is obtained.



Example:

Given, Gray code is 110101.

Step 1 − Keep the MSB of the Gray code as it is like 1 in this case.

Step 2 − By comparing successive two bits diagonally of Gray code, we will get the Binary code as 100110.

Exercise: Find the Binary code of the following numbers-

(i) 101011 (ii) 1100110