2

CHAPTER

NUMBER SYSTEM

2.1 Numbering Systems

LONG QUESTIONS

Q. 21 Define Number System. Explain the Decimal, Binary, Octal, and Hexadecimal Number Systems with Examples.

Ans: Numbering systems are used to represent numbers in different formats, essential for computing and digital electronics. The primary numbering systems are decimal, binary, octal, and hexadecimal.

1. Decimal System (Base 10): The decimal system is the standard system used by humans, consisting of ten digits: 0 to 9. Each digit represents a power of 10. For example:
   $523 = 5 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 = 500 + 20 + 3 = 523$.
2. Binary System (Base 2): Binary uses only two digits: 0 and 1. It is fundamental in computing because digital circuits have two states: on (1) and off (0). Each binary digit represents a power of
   For example:
   Binary $1011 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 8 + 0 + 2 + 1 = 11$ (in decimal).
3. Octal System (Base 8): The octal system uses digits 0 to 7. It is related to binary because each octal digit represents three binary digits. For example:
   Octal $157 = 1 \times 8^2 + 5 \times 8^1 + 7 \times 8^0 = 64 + 40 + 7 = 111$ (in decimal).
4. Hexadecimal System (Base 16): Hexadecimal uses digits 0-9 and letters A-F, representing values from 0 to 15. Each hexadecimal digit corresponds to four binary digits. For example:
   Hexadecimal $1A3 = 1 \times 16^2 + A (10) \times 16^1 + 3 \times 16^0 = 256 + 160 + 3 = 419$ (in decimal).

   Conclusion: Each of these systems serves a unique purpose in computing. Decimal is the most familiar, binary is used by computers, octal and hexadecimal offer more compact representations of binary numbers.

Q. 1 Describe the Conversion Techniques Between Decimal, Binary, Octal, and Hexadecimal Systems. Provide Detailed Examples.

Ans: Conversion between different numbering systems is essential for understanding and working with data in computers. Below are the conversion techniques with detailed examples:

1. Decimal to Binary:

31. Divide the decimal number by 2 and record the remainders.
32. Continue dividing the quotient until it becomes 0.
33. Read the remainders from bottom to top.

Example:

Convert 83 to binary:
$83 \div 2 = 41$, remainder 1
$41 \div 2 = 20$, remainder 1
$20 \div 2 = 10$, remainder 0
$10 \div 2 = 5$, remainder 0
$5 \div 2 = 2$, remainder 1
$2 \div 2 = 1$, remainder 0
$1 \div 2 = 0$, remainder 1
Reading the remainders bottom to top: 1010011 (binary).

2. **Decimal to Octal:**

21. Divide the decimal number by 8 and record the remainders.
22. Continue dividing the quotient until it becomes 0.
23. Read the remainders from bottom to top.

Example: Convert 83 to octal:

24. 83 ÷ 8 = 10, remainder 3
25. 10 ÷ 8 = 1, remainder 2
26. 1 ÷ 8 = 0, remainder 1

Reading the remainders bottom to top: 123 (octal).

3. **Decimal to Hexadecimal:**

27. Divide the decimal number by 16 and record the remainders.
28. Continue dividing the quotient until it becomes 0.
29. Read the remainders from bottom to top.

Example:
Convert 2297 to hexadecimal:

30. 2297 ÷ 16 = 143, remainder 9
31. 143 ÷ 16 = 8, remainder F
32. 8 ÷ 16 = 0, remainder 8

Reading the remainders bottom to top: 8F9 (hexadecimal).

4. **Binary to Octal:**

33. Group the binary number into sets of three bits from right to left.
34. Convert each group into its corresponding octal digit.

Example:
Convert binary 110101011 to octal: Grouping the binary number into sets of three: 110 101 011.

35. 110 = 6
36. 101 = 5
37. 011 = 3 Result: 653 (octal).

5. **Binary to Hexadecimal:**

38. Group the binary number into sets of four bits from right to left.
39. Convert each group into its corresponding hexadecimal digit.

Example:
Convert binary 1101011010110010 to hexadecimal: Grouping the binary number into sets of four: 1101 0110 1011 0010.

40. 1101 = D
41. 0110 = 6
42. 1011 = B
43. 0010 = 2 Result: D6B2 (hexadecimal).

Hexadecimal	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Table2.2: Correspondence between hexadecimal and binary digits

SHORT QUESTIONS

Q. 1 What is a numbering system?

Ans:

NUMBERING SYSTEM

A numbering system is a way of representing numbers using specific symbols or digits. It is the foundation of data representation in computers and digital electronics. Common numbering systems include decimal, binary, octal, and hexadecimal, each with a different base (10, 2, 8, and 16, respectively).

Q. 2 What is the decimal number system?

Ans:

DECIMAL NUMBER SYSTEM

The decimal system is a base-10 numbering system that uses digits from 0 to 9. Each digit in a number represents a power of 10 based on its position. For example, in the number 523, the '5' represents 5×10^2, the '2' represents 2×10^1, and the '3' represents 3×10^0.

Q. 3 How does binary representation work in computers?

Ans:

BINARY REPRESENTATION WORK IN COMPUTERS

Binary is a base-2 system, meaning it uses only two digits: 0 and 1. These digits are used to represent all types of data in computers, such as numbers, text, and images. The binary system fits well with digital circuits, as they have two states: on (1) and off (0).

Q. 4 Explain the conversion from decimal to binary.

Ans:

CONVERSION FROM DECIMAL TO BINARY

To convert a decimal number to binary, divide the number by 2 and write down the remainder. Repeat the division process for the quotient until it reaches zero. The binary equivalent is read from the bottom to the top of the remainders.

Q. 5 What is the octal number system?

Ans:

OCTAL NUMBER SYSTEM

The octal number system is a base-8 system that uses digits from 0 to 7. Each digit represents a power of 8. The octal system is related to binary because every octal digit corresponds to a 3-bit binary number, making it easy to convert between the two systems.

Q. 6 How do you convert a decimal number to octal?

Ans:

DECIMAL NUMBER TO OCTAL

To convert a decimal number to octal, divide the decimal number by 8, recording the remainder each time. Continue the division until the quotient is zero. The octal number is the sequence of remainders read from bottom to top.

Q. 7 What is the hexadecimal number system?

Ans:

HEXADECIMAL NUMBER SYSTEM

The hexadecimal system is a base-16 system that uses digits 0–9 and letters A–F, where A represents 10, B represents 11, and so on up to F, which represents 15. Each hexadecimal digit represents four binary digits (bits), making it a compact way of representing large binary numbers.

Q. 8 Explain the conversion from decimal to hexadecimal.

Ans:

CONVERSION FROM DECIMAL TO HEXADECIMAL

To convert a decimal number to hexadecimal, divide the number by 16, recording the quotient and the remainder. Continue dividing the quotient by 16 until it reaches zero. The hexadecimal result is the sequence of remainders read from bottom to top.

Q. 9 What is ASCII?

Ans:

ASCII

ASCII (American Standard Code for Information Interchange) is a text encoding scheme that represents characters using 7 or 8 bits. It includes standard English characters, numbers, and punctuation. Each character in ASCII is assigned a unique numeric value.

Q. 10 What is Unicode?

Ans: **UNICODE**

Unicode is an extended text encoding system that can represent characters from virtually every written language in the world. Unlike ASCII, which is limited to 128 characters, Unicode can handle a vast range of characters from multiple languages and symbols.

Q. 11 How is machine-level data represented?

Ans: **MACHINE-LEVEL DATA REPRESENTED**

Machine-level data is represented using binary code. In computers, all types of data, including numbers, text, and images, are ultimately stored and processed in binary format. Each instruction is converted into a binary sequence that the CPU can interpret and execute.

Q. 12 What is binary arithmetic?

Ans: **BINARY ARITHMETIC**

Binary arithmetic involves performing basic arithmetic operations (addition, subtraction, multiplication, division) on binary numbers. The process follows the same principles as decimal arithmetic but is based on powers of 2.

Q. 13 How do computers store images, audio, and video?

Ans: **COMPUTERS STORE IMAGES, AUDIO, AND VIDEO**

Images, audio, and video are stored as binary data. Each element (pixel in an image, sample in audio, or frame in video) is converted into binary code, with specific encoding schemes to ensure the data is accurately represented and can be reconstructed for display or playback.

Q. 14 What are file formats and extensions?

Ans: **FORMATS AND EXTENSIONS**

File formats define how data is stored in a file, and file extensions are used to identify the format. For example, .txt denotes a text file, .jpg denotes an image file, and .mp3 denotes an audio file. Each format has specific rules for storing data.

Q. 15 What is binary data manipulation?

Ans: **BINARY DATA MANIPULATION**

Binary data manipulation involves operations such as shifting, masking, or logical operations (AND, OR, NOT) on binary data. These operations are essential for low-level data processing and are widely used in digital electronics and computer programming.

Q. 16 Why is the binary system used in computers?

Ans: **BINARY SYSTEM USED IN COMPUTERS**

The binary system is used because digital circuits have two states: on (1) and off (0). These two states can be easily represented using binary digits, making it ideal for computers that rely on electrical signals to perform computations.

Q. 17 What is the difference between octal and binary systems?

Ans: **DIFFERENCE BETWEEN OCTAL AND BINARY SYSTEMS**

The octal system is base-8, using digits from 0 to 7, while the binary system is base-2, using only 0 and 1. Each octal digit corresponds to three binary digits, making octal a more compact representation of binary numbers.

Q. 18 What is the relationship between hexadecimal and binary systems?

Ans: **RELATIONSHIP BETWEEN HEXADECIMAL AND BINARY SYSTEMS**

The hexadecimal system is base-16, while the binary system is base-2. Each hexadecimal digit corresponds to four binary digits (bits), making it easier to convert between the two systems.

Q. 19 How do you represent real numbers in binary?

Ans: **REPRESENT REAL NUMBERS IN BINARY**

Real numbers are represented in binary using floating-point notation, where the number is divided into a mantissa (the significant digits) and an exponent (the power of 2). This allows for the representation of very large or very small numbers.

Q. 20 What is a binary to octal conversion?

Ans: **BINARY TO OCTAL CONVERSION**

Binary to octal conversion involves grouping the binary digits in groups of three from right to left. Each group is then converted into its corresponding octal digit.

Q. 21 What is a binary to hexadecimal conversion?

Ans: **Binary To Hexadecimal Conversion**

Binary to hexadecimal conversion involves grouping the binary digits in groups of four from right to left. Each group is then converted into its corresponding hexadecimal digit.

Q. 22 What are binary arithmetic operations?

Ans: **BINARY ARITHMETIC OPERATIONS**

Binary arithmetic operations include addition, subtraction, multiplication, and division. These operations are performed similarly to decimal arithmetic but use binary rules, where only the digits 0 and 1 are involved.

Q. 23 What is the role of encoding schemes in data representation?

Ans: **ROLE OF ENCODING SCHEMES IN DATA REPRESENTATION**

Encoding schemes like ASCII and Unicode convert data into a format that can be easily processed by computers. These schemes assign unique binary values to characters and symbols, ensuring that data is stored and transferred correctly.

Q. 24 What is the significance of file extensions?

Ans: **SIGNIFICANCE OF FILE EXTENSIONS**

File extensions help identify the type of data contained in a file. They ensure that the correct software or application is used to open and process the file, such as .mp3 for audio or .jpg for images.

Q. 25 What is the importance of numbering systems in computing?

Ans: **IMPORTANCE OF NUMBERING SYSTEMS IN COMPUTING**

Numbering systems are crucial in computing because they provide the foundation for representing all kinds of data in digital form. Computers use binary to store and process data, and conversion between different numbering systems allows efficient data storage and manipulation.

MULTIPLE CHOICE QUESTIONS

- **What is the base of the decimal system?**
  (A) 8 (B) 10
  (C) 16 (D) 2
- **In the binary number system, what is the base?**
  (A) 8 (B) 10
  (C) 2 (D) 16
- **Which of the following is NOT a binary digit?**
  (A) 1 (B) 0
  (C) 2 (D) Both a and b
- **How is the number 523 represented in the decimal system?**
  (A) $5 \times 10^2 + 2 \times 10^1 + 3 \times 10^0$ (B) $5 \times 10^2 + 2 \times 10^0 + 3 \times 10^1$
  (C) $5 + 2 + 3$ (D) None of the above
- **What is the binary representation of decimal number 83?**
  (A) 1001011 (B) 1010011
  (C) 1101011 (D) 1110101
- **Which numbering system is used by computers at the lowest level?**
  (A) Octal (B) Decimal
  (C) Binary (D) Hexadecimal
- **What is the base of the hexadecimal number system?**
  (A) 10 (B) 2
  (C) 8 (D) 16

- **In the binary system, the number 1011 equals which decimal value?**
  (A) 10 (B) 13
  (C) 11 (D) 15
- **What is the place value of the second digit from the right in the binary number system?**
  (A) 2^1 (B) 2^0
  (C) 2^2 (D) 2^3
- **Which number system is base-8?**
  (A) Binary (B) Decimal
  (C) Octal (D) Hexadecimal
- **How is the number 157 represented in the octal system?**
  (A) 11110 (B) 1111
  (C) 152 (D) 11101
- **What is the octal equivalent of the binary number 110101011?**
  (A) 653 (B) 703
  (C) 753 (D) 613
- **Which system is easier to read large binary numbers?**
  (A) Decimal (B) Hexadecimal
  (C) Octal (D) Binary
- **The letter A in the hexadecimal system represents which decimal value?**
  (A) 10 (B) 11
  (C) 12 (D) 15
- **Which of the following is used in hexadecimal numbers to represent values greater than 9?**
  (A) A-F (B) 0-9
  (C) 0-7 (D) X-Y
- **What is the hexadecimal equivalent of the binary number 1101011010110010?**
  (A) D6B2 (B) C5A1
  (C) A5F2 (D) B6C1
- **What is the base of the octal system?**
  (A) 2 (B) 8
  (C) 10 (D) 16
- **Which system is used for representing values of memory locations in computers?**
  (A) Binary (B) Decimal
  (C) Hexadecimal (D) Octal
- **In the binary number system, which digit represents the "off" state?**
  (A) 1 (B) 0
  (C) A (D) F
- **What is the place value of the third digit from the right in the decimal number system?**
  (A) 10^2 (B) 10^3
  (C) 10^0 (D) 10^1
- **How many digits are there in the octal system?**
  (A) 8 (B) 10
  (C) 16 (D) 2
- **Which system is used to store colors in modern computers?**
  (A) Binary (B) Decimal
  (C) Hexadecimal (D) Octal
- **What is the hexadecimal equivalent of decimal 2297?**
  (A) 8F9 (B) 9F8
  (C) 7F9 (D) 8D9
- **The conversion of 83 from decimal to binary gives which result?**
  (A) 1010111 (B) 1101011
  (C) 1010011 (D) 1001111

- **Which of the following is used to represent the binary number 1101?**
  (A) D (B) C
  (C) E (D) F
- **What is the primary use of the binary system in computers?**
  (A) Text encoding (B) Data transfer
  (C) Data storage (D) All of the above
- **What does ASCII stand for?**
  (A) American Standard Code for Information Interchange
  (B) Advanced System for Code Integration
  (C) American Standard Code for Information Interface
  (D) All of the above
- **In which system do each digit represent four binary bits?**
  (A) Binary (B) Hexadecimal
  (C) Octal (D) Decimal
- **Which of the following is NOT a commonly used text encoding scheme?**
  (A) ASCII (B) Unicode
  (C) Hexadecimal (D) None of the above
- **What is the purpose of file extensions?**
  (A) To identify the type of file (B) To help the system open the file
  (C) To associate a file with a program (D) All of the above

2.2 Data Representation in Computing Systems

LONG QUESTIONS

- **Explain the process of converting a decimal number to its two's complement binary representation. Provide an example.**

Ans: To convert a decimal number into its two's complement binary representation, the following steps are involved:

1. Start with the binary representation of the positive version of the decimal number.

Example: Convert -5 into 8-bit two's complement.
First, write the binary of +5: 00000101.

Invert all the bits.

Change every 0 to 1 and every 1 to 0. The inversion of 00000101 is 11111010.

Add 1 to the inverted bits.

Add 1 to the least significant bit of the inverted binary number:
11111010 + 1 = 11111011
So, the two's complement binary representation of -5 in 8-bit form is 11111011.
This method allows for the representation of negative numbers in a way that simplifies binary arithmetic in computers.

SHORT QUESTIONS

Q. 22 What is a whole number in computing?

Ans: **WHOLE NUMBER IN COMPUTING**

Whole numbers are non-negative integers that include zero and all positive integers. They are used in computing to represent quantities that cannot be negative, such as the number of students in a school, a person's age, or grades. For example, a student's age in years (e.g., 10) is represented as a whole number in a computer.

Q. 23 How is a 1-byte integer represented in binary?

Ans: **1-BYTE INTEGER REPRESENTATION IN BINARY**

A 1-byte integer is made up of 8 bits. The maximum value of a 1-byte integer is represented as 11111111 in binary, which equals 255 in decimal. The minimum value is represented as

00000000, which equals 0. The binary representation allows computers to store values ranging from 0 to 255 in 1 byte.

Q. 24 How are negative values represented in computers?

Ans: **<u>NEGATIVE VALUES REPRESENTED IN COMPUTERS</u>**

Negative values are represented using two's complement in binary. In this method, the sign bit (most significant bit) indicates if the value is negative. For example, the decimal number -5 is represented as 11111011 in 8-bit two's complement after inverting all the bits of 5 and adding 1 to the least significant bit.

Q. 25 What is the two's complement method?

Ans: **<u>TWO'S COMPLEMENT METHOD</u>**

Two's complement is a method used to represent negative integers in binary. To find the two's complement of a number, invert all the bits and add 1 to the least significant bit. For example, the binary representation of 5 is 00000101. To get -5, we invert all the bits to get 11111010 and then add 1 to get 11111011.

Q. 26 What is the significance of the sign bit in signed integers?

Ans: **<u>SIGNIFICANCE OF THE SIGN BIT IN SIGNED INTEGERS</u>**

In signed integers, one bit (usually the most significant bit) is reserved as the sign bit. If the sign bit is 1, the number is negative; if the sign bit is 0, the number is positive. For example, in an 8-bit signed integer, the number 127 is represented as 01111111 (sign bit is 0), while -128 is represented as 10000000 (sign bit is 1).

Q. 27 What is the maximum value for a 2-byte signed integer?

Ans: **<u>MAXIMUM VALUE FOR A 2-BYTE SIGNED INTEGER</u>**

A 2-byte signed integer uses 16 bits, with the first bit as the sign bit. The maximum value is represented by the binary number 0111111111111111, which equals 32,767 in decimal. This is because 2^15 - 1 equals 32,767.

Q. 28 What is the minimum value for a 4-byte signed integer?

Ans: **<u>MINIMUM VALUE FOR A 4-BYTE SIGNED INTEGER</u>**

A 4-byte signed integer uses 32 bits, with the first bit as the sign bit. The minimum value is represented by 10000000000000000000000000000000, which equals -2,147,483,648 in decimal. This is computed as -2^31, where 31 is the number of bits reserved for the value.

Q. 29 How is the minimum value of a signed integer calculated?

Ans: **<u>MINIMUM VALUE OF A SIGNED INTEGER CALCULATED</u>**

The minimum value of a signed integer is calculated using the formula -2^(n-1), where n is the number of bits. For example, in a 1-byte signed integer (8 bits), the minimum value is -2^7 = -128. This method ensures that negative values are represented using two's complement.

Q. 30 What is the role of two's complement in negative number representation?

Ans: **<u>ROLE OF TWO'S COMPLEMENT IN NEGATIVE NUMBER REPRESENTATION</u>**

Two's complement allows computers to store negative numbers efficiently by inverting the bits of a positive number and adding 1. This method simplifies binary arithmetic by making addition and subtraction operations consistent for both positive and negative numbers.

Q. 31 How do you calculate the maximum value for a signed integer in n bits?

Ans: **<u>CALCULATE THE MAXIMUM VALUE FOR A SIGNED INTEGER IN N BITS</u>**

The maximum value for a signed integer in n bits is calculated using the formula 2^(n-1) - 1. This formula accounts for the sign bit and ensures that the range of values is balanced between positive and negative numbers. For example, for a 16-bit signed integer, the maximum value is 2^15 - 1 = 32,767.

Q. 32 What is an example of a whole number represented in computing?

Ans: **<u>EXAMPLE OF A WHOLE NUMBER REPRESENTED IN COMPUTING</u>**

A whole number, like the number of students in a class (e.g., 30), is represented in computing using a non-negative integer. Since it cannot be negative, it is stored as a whole number in memory, typically using 1 byte for small values, such as 30.

Q. 33 How are larger integer values stored in memory?

Ans: **LARGER INTEGER VALUES STORED IN MEMORY**

Larger integer values are stored in memory using more bytes. A 2-byte integer (16 bits) can store values up to 65,535, while a 4-byte integer (32 bits) can store values up to 4,294,967,295. By using more bits, computers can handle larger values in their computations.

Q. 34 What happens when all the bits in a 1-byte signed integer are set to 1?

Ans: **BITS IN A 1-BYTE SIGNED INTEGER ARE SET TO 1**

When all the bits in a 1-byte signed integer are set to 1, the value represented is -128. This is because the sign bit is 1, indicating a negative value, and the two's complement method is used to find the negative value.

Q. 35 Why is two's complement used for negative numbers in computers?

Ans: **NEGATIVE NUMBERS IN COMPUTERS**

Two's complement is used because it simplifies the representation of negative numbers and allows easy addition and subtraction of signed integers. The system eliminates the need for separate handling of positive and negative numbers in binary arithmetic.

Q. 36 How is the maximum value for a 4-byte signed integer computed?

Ans: **MAXIMUM VALUE FOR A 4-BYTE SIGNED INTEGER**

The maximum value for a 4-byte signed integer is computed as 2^(32-1) - 1 = 2^31 - 1 = 2,147,483,647. This is because 31 bits are available to store the value, with one bit reserved for the sign. This allows for a large range of positive values.

MULTIPLE CHOICE QUESTIONS

Q.16 What is the maximum value of a 1-byte whole number?
(A) 128 (B) 255
(C) 65,535 (D) 4,294,967,295

Q.17 Which of the following is the set of whole numbers?
(A) Z = {..., -3, -2, -1, 0, 1, 2, 3, ...} (B) W = {0, 1, 2, 3, ...}
(C) W = {..., -3, -2, -1, 0} (D) None of the above

Q.18 How are negative integers represented in computers?
(A) Using two's complement (B) Using sign magnitude
(C) Using absolute values (D) Using unsigned integers

Q.19 What is the maximum value that can be represented by a 2-byte whole number?
(A) 255 (B) 65,535
(C) 127 (D) 32,768

Q.20 In a 1-byte signed integer, which bit is used as the sign bit?
(A) Least Significant Bit (LS(B) (B) Most Significant Bit (MS(B)
(C) First bit from the left (D) First bit from the right

Q.21 What is the maximum positive value for a 1-byte signed integer?
(A) 127 (B) 255
(C) 128 (D) 64,000

Q.22 Which of the following is the method used to represent negative values in computers?
(A) Two's complement (B) Sign magnitude
(C) Binary representation (D) None of the above

Q.23 How do you convert a decimal number into its two's complement form?
(A) Add 1 to the number
(B) Invert all the bits and add 1 to the least significant bit
(C) Simply invert the bits
(D) Divide the number by 2

Q.24 What is the binary representation of -5 in 8-bit two's complement?
(A) 00000101 (B) 11111011
(C) 10000101 (D) 11111111

Q.25 For a 4-byte signed integer, what is the minimum value it can represent?
(A) -128 (B) -32,768
(C) -2,147,483,648 (D) -4,294,967,295

Q.26 What does the term "two's complement" refer to?
(A) A method of encoding unsigned integers
(B) A method to handle signed integers and negative values
(C) A way to represent characters in ASCII
(D) A way to store floating-point numbers

Q.27 What is the maximum value that can be represented by a 4-byte signed integer?
(A) 4,294,967,295 (B) 2,147,483,647
(C) 32,768 (D) 65,535

Q.28 In a 1-byte signed integer, what would the bit pattern 10000000 represent?
(A) 128 (B) -128
(C) 255 (D) -1

Q.29 Which value does an 8-bit whole number with all bits set to 0 represent?
(A) 0 (B) -128
(C) 255 (D) 1

Q.30 What is the formula used to calculate the maximum value in an n-bit integer?
(A) 2^n (B) 2^n - 1
(C) 2^(n-1) (D) 2^(n+1)

2.3 STORING REAL VALUES IN COMPUTER MEMORY

LONG QUESTION

Q. 1 Explain the process of converting a real number into its binary representation, specifically focusing on the fractional part. Include an example of converting 0.375.

Ans: Identifying the Fractional Part

First, identify the fractional part of the real number. For example, in the number 4.625, the fractional part is 0.625.

Converting the Fractional Part to Binary

Multiply the fractional part by 2, and write down the integer part of the result. Repeat this process until the fractional part becomes zero or the desired precision is achieved.

Example: Converting 0.375

Q.11 $0.375 \times 2 = 0.75 \rightarrow$ Integer part: 0
Q.12 $0.75 \times 2 = 1.5 \rightarrow$ Integer part: 1
Q.13 $0.5 \times 2 = 1.0 \rightarrow$ Integer part: 1

The binary representation of 0.375 is 0.011.

Combining the Integer and Fractional Parts

After converting both the integer and fractional parts, combine them to get the complete binary representation of the real number. In this case, 4.625 would be represented as:
Integer part (4) in binary: 100
Fractional part (0.625) in binary: 0.101
So, 4.625 in binary is 100.101.

Q. 2 How does the computer stores Real Values in Computer Memory? Explain Single Precision and Double Precision Methods in detail.

Ans: Storing Real Values in Computer Memory

In computers, real values, also known as floating-point numbers, are used to represent number with fractions and/or decimals.

Understanding Floating-Point Representation

Floating-point numbers (real values) are represented similarly to scientific notation as given below:

A floating-point number = sign x mantissa x 2 porent, according to the above formula, 5.75 is represented as 1.4375 x 2. To convert the fractional part of a real (floating-point) number from decimal (base-10) to binary (base-2), multiply the fractional part by 2 and write down the integral part of the result. Repeat this process with the new fractional part until the value of the fractional part becomes zero or until the required precision is achieved.

Single Precision (32-bit)

In this standard, 4 bytes (or 32 bits) are assigned where the 1st bit is the sign bit, and the next 8 bits are for the exponent and the remaining 23 bits are for the mantissa.

Here the exponent can be ranged between -126 and + 127.

The approximate range of values from1.4 x 10 to 3.4 x 10.

Value	Representation	Sign Bit	Exponent (8 bits)	Mantissa (23 bits)
Grouping		1 bit	8 bits	23 bits
5.75	1.4375×2^2	0	10000001	1011100000000000000000
-5.75	$- 1.4375 \times 2^2$	1	10000001	1011100000000000000000
0.15625	1.25×2^{-3}	0	01111101	0100000000000000000000
-0.15625	-1.25×2^{-3}	1	01111101	0100000000000000000000

Table 2.3: 32-bit Floating Point Representation

Explanation:

Each floating point value is broken down into three main components: the sign bit, the exponent, and the mantissa.

Grouping: This row explains the bit allocation for the 32-bit floating point format: 1 bit for the sign, 8 bits for the exponent, and 23 bits for the mantissa.

1. 5.75: Representation: 1.4375 x 2 - Sign Bit: 0 (positive) - Exponent: 2 +127 = 129, which is 10000001, Mantissa: The binary representation of 0.4375 is

2. -5.75: Representation: -1.4375×2^2 - Sign Bit: 1 (negative) - Exponent: 2 + 127 = 129, which is 10000001, - Mantissa: The binary representation of 0.4375 is

3. 0.15625: Representation: 1.25×2^3 - Sign Bit: 0 (positive) - Exponent: -3 +127 = 124, which is 01111101, Mantissa: The binary representation of 0.25 is

4. -0.15625: Representation: -1.25 x 2 - Sign Bit: 1 (negative) - Exponent: -3 +127 = 124, which is 01111101, - Mantissa: The binary representation of 0.25

This breakdown helps illustrate how floating point values are stored and manipulated in computer systems.

Double Precision (64-bit)

In double precision, the exponent is represented using 11 bits. The exponent is stored in a biased form, with a bias of 1023. The range of the actual exponent values can be determined as follows:

• Bias: 1023

• Exponent range: The actual exponent values range from-1022 to +1023. Therefore, the smallest and largest possible exponent values in double- precision are:

• Minimum exponent: -1022

• Maximum exponent: +1023

We can perform the same steps given for the single-precision, except the difference of the abovementioned values.

SHORT QUESTIONS

Q. 1 Explain how the fractional part of a number is converted to binary.

Ans: **FRACTIONAL PART OF A NUMBER IS CONVERTED TO BINARY**

To convert the fractional part of a decimal number to binary, multiply the fractional part by 2 and write down the integer part of the result. Repeat the process with the new fractional part until it becomes zero or the desired precision is achieved. For example, to convert 0.375 to binary:

- $0.375 \times 2 = 0.75 \rightarrow$ Integer part: 0
- $0.75 \times 2 = 1.5 \rightarrow$ Integer part: 1
- $0.5 \times 2 = 1.0 \rightarrow$ Integer part: 1

The result is 0.011 in binary.

Q. 2 What is the formula for representing a floating-point number?

Ans: **<u>REPRESENTING A FLOATING-POINT NUMBER</u>**

The formula for representing a floating-point number is:
Floating-point number = sign × mantissa × 2^exponent.
This allows the representation of real numbers as a combination of a sign, a mantissa, and an exponent.

Q. 3 What is the purpose of the exponent in floating-point representation?

Ans: **<u>PURPOSE OF THE EXPONENT IN FLOATING-POINT REPRESENTATION</u>**

The exponent in floating-point representation determines the scale of the number. It adjusts the position of the decimal point (or binary point) in the mantissa to represent larger or smaller values. It allows the representation of a wide range of values.

Q. 4 How is the exponent stored in single precision?

Ans: **<u>EXPONENT STORED IN SINGLE PRECISION</u>**

In single precision, the exponent is stored in 8 bits, and the range of the exponent is from -126 to +127. The exponent is stored in a biased form with a bias of 127.

Q. 5 What is the range of the exponent in double precision?

Ans: **<u>RANGE OF THE EXPONENT IN DOUBLE PRECISION</u>**

In double precision, the exponent is stored in 11 bits with a bias of 1023. The range of the actual exponent values is from -1022 to +1023.

Q. 6 How are negative values represented in floating-point numbers?

Ans: **<u>NEGATIVE VALUES REPRESENTED IN FLOATING-POINT NUMBERS</u>**

Negative values in floating-point representation are indicated by the sign bit. If the sign bit is 1, the number is negative. This is true for both single and double precision floating-point numbers.

Q. 7 Explain the significance of the mantissa in floating-point representation.

Ans: **<u>SIGNIFICANCE OF THE MANTISSA IN FLOATING-POINT REPRESENTATION</u>**

The mantissa represents the significant digits of the floating-point number. It is the fractional part of the number, which is converted to binary and multiplied by 2 raised to the power of the exponent.

Q. 8 What is the role of the sign bit in a 32-bit floating-point number?

Ans: **<u>ROLE OF THE SIGN BIT IN A 32-BIT FLOATING-POINT NUMBER</u>**

The sign bit in a 32-bit floating-point number determines whether the number is positive or negative. If the sign bit is 0, the number is positive; if it is 1, the number is negative.

Q. 9 Explain the process of converting the number 5.75 into floating-point representation.

Ans: To convert 5.75 into floating-point representation:
Step 1: Convert 5.75 to binary. The binary of 5 is 101 and 0.75 is 0.11.
Step 2: Express it in scientific notation: 1.4375 × 2^2.
Step 3: The sign bit is 0 (positive), the exponent is 2 + 127 = 129 (which is 10000001 in binary), and the mantissa is the binary of 0.4375, which is 0.011.
Final representation: 0 | 10000001 | 01111000000000000000000.

Q. 10 What happens when a number is represented using double precision (64-bit)?

Ans: **<u>REPRESENTED USING DOUBLE PRECISION (64-BIT)</u>**

In double precision, the number is represented using 64 bits, where the sign bit is 1 bit, the exponent is 11 bits, and the mantissa is 52 bits. Double precision allows for greater precision and a wider range of values compared to single precision.

Q. 11 How does the biasing of the exponent affect floating-point representation?

Ans: **BIASING OF THE EXPONENT**

Biasing the exponent in floating-point representation allows both positive and negative exponents to be represented as unsigned integers. The exponent is adjusted by adding the bias (127 for single precision and 1023 for double precision), which helps handle a wide range of values.

Q. 12 What is the smallest positive number that can be represented in double precision?

Ans: **REPRESENTED IN DOUBLE PRECISION**

The smallest positive number representable in double precision is approximately 4.9×10^{-32}, which is much smaller than the smallest representable number in single precision.

Q. 13 Why are floating-point numbers used in computers?

Ans: **FLOATING-POINT NUMBERS USED IN COMPUTERS**

Floating-point numbers are used to represent real values with fractional parts, such as decimals, in computers. They are essential for storing and processing numbers with large ranges and precision.

Q. 14 What is the result of converting the number 0.15625 to binary?

Ans: **CONVERTING THE NUMBER 0.15625 TO BINARY**

To convert 0.15625 to binary:

$0.15625 \times 2 = 0.3125 \rightarrow$ Integer part: 0

$0.3125 \times 2 = 0.625 \rightarrow$ Integer part: 0

$0.625 \times 2 = 1.25 \rightarrow$ Integer part: 1

$0.25 \times 2 = 0.5 \rightarrow$ Integer part: 0

$0.5 \times 2 = 1.0 \rightarrow$ Integer part: 1

The binary representation is 0.00101.

Q. 15 What is the bias used in double precision exponent representation?

Ans: **DOUBLE PRECISION EXPONENT REPRESENTATION**

The bias used in double precision exponent representation is 1023. This bias is added to the actual exponent to store it as an unsigned value.

MULTIPLE CHOICE QUESTIONS

21. What is the formula for representing a floating-point number?

(A) sign × exponent × mantissa (B) sign × mantissa × 2^exponent
(C) sign × mantissa × exponent^2 (D) sign × exponent × 2^mantissa

22. What is the fractional part of 4.625?

(A) 0.625 (B) 0.75
(C) 0.25 (D) 4.625

23. What is the first step in converting a fractional decimal number to binary?

(A) Identify the sign bit (B) Convert the fractional part to binary
(C) Identify the fractional part (D) Divide by 2

24. How many bits are assigned to the exponent in a 32-bit floating-point number?

(A) 8 bits (B) 16 bits
(C) 23 bits (D) 4 bits

25. What is the range of the exponent in a 32-bit floating-point number?

(A) -255 to 255 (B) -126 to 127
(C) -128 to 128 (D) -10 to 10

26. What is the binary representation of the integer part 5 in 0.375?

(A) 101 (B) 100
(C) 0 (D) 111

27. What is the result of converting 0.375 to binary?

(A) 0.01 (B) 0.1
(C) 0.011 (D) 0.101

28. How many bits are allocated to the mantissa in single precision (32-bit) representation?
(A) 8 bits (B) 16 bits
(C) 23 bits (D) 32 bits

29. What is the exponent range for double precision (64-bit) representation?
(A) -1023 to 1023 (B) -1022 to 1023
(C) -127 to 127 (D) -255 to 255

30. In double precision, how many bits are assigned to the exponent?
(A) 8 bits (B) 16 bits
(C) 23 bits (D) 11 bits

31. What is the smallest positive number representable in single precision?
(A) 1.4×10^{-45} (B) 1.4×10^{18}
(C) 1.4×10^{45} (D) 1.4×10^{-45}

32. What is the range of values for the exponent in single precision (32-bit)?
(A) -255 to 255 (B) -1022 to 1023
(C) -126 to 127 (D) -5 to 5

33. What is the primary purpose of the sign bit in floating-point representation?
(A) To represent the magnitude of the number
(B) To determine whether the number is positive or negative
(C) To represent the fractional part
(D) To represent the exponent

34. How do you calculate the mantissa in floating-point representation?
(A) By converting the integer part to binary (B) By converting the fractional part to binary
(C) By multiplying the decimal part by 2 (D) By adding the integer and fractional parts

35. What is the smallest positive number representable in double precision?
(A) 1.4×10^{45} (B) 4.9×10^{-32}
(C) 4.9×10^{-32} (D) 4.9×10^{-32}

2.4 BINARY ARITHMETIC OPERATIONS

LONG QUESTION

Q. 3 Explain Binary Arithmetic and provide example for each operation.

Ans: Binary Arithmetic Operations

Arithmetic operations include addition, subtraction, multiplication and division, and are performed on two numbers at a time. Binary arithmetic operations are similar to decimal operations but follow binary rules. Here's a brief overview of the basic operations:

1. Addition

Binary addition uses only two digits: 0 and 1. Here, we will learn how to add binary numbers and how to handle the addition of negative binary numbers.

Binary Addition Rules

Binary addition follows these simple rules:

1.0+0=0.
2.0+1=1
3.1+0=1
4. 1+1=0 (with a carry of 1 to the next higher bit)

Example of Binary Addition

Example 1:

1101
+1011
11000

In this example:

1+1=0 (carry 1)
0+1+1 (carry) = 0 (carry 1)

1+1+1 (carry) = 1 (carry 1)
1+0+1 (carry) = 0 (carry 1)

2. Subtraction

In binary arithmetic, subtraction can also be carried out by adding the two's complement or the value of the subtrahend to the minuend.

Example: Subtract 6 from 9 in Binary Minuend = 9 = 1001,
Subtrahend = 6 = 0110

Step 1: Find the Two's Complement of the Subtrahend

Invert the bits of 0110: Inversion: 1001
Add 1 to the inverted number: 1001+1=1010 =-6

Step 2: Add the Minuend and the Two's Complement of the Subtrahend

1001+1010=10011,
Step 3: Discard the Carry Bit 10011, Discard carry, 0011=3
So, 9-6=3.

3. Multiplication

Binary numbers are base-2 numbers, consisting of only Os and 1s. Multiplying binary numbers follows similar principles to multiplying decimal numbers, but with simpler rules. Here, we will learn how to multiply binary numbers with example.

Steps to Multiply Binary Numbers

1. Write down the binary numbers, aligning them by the least significant bit (rightmost bit).
2. Multiply each bit of the second number by each bit of the first number, similar to the long multiplication method in decimal.
3. Shift the partial results one place to the left for each new row, starting from the second row.
4. Add all the partial results to get the final product.

Example

Let's multiply two binary numbers:
101 (binary for 5)
× 11 (binary for 3)
--------- 101 (101 × 1, rightmost bit of 11)
+ 1010 (101 × 1, shifted one place to the left for the second bit of 11)
--------- 1111 (Final result in binary)

4. Division

Binary division is similar to decimal division but only involves two digits: 0 and 1. It follows steps like comparing, subtracting, and shifting, akin to long division in the decimal system.

Steps of Binary Division

1. Compare: Compare the divisor with the current portion of the dividend.
2. Subtract: Subtract the divisor from the dividend portion if the divisor is less than or equal to the dividend.
3. Shift: Shift the next binary digit from the dividend down to the remainder.
4. Repeat: Repeat the process until all digits of the dividend have been used.

Example:

```
110 (Quotient in binary)
-------
10 | 1100
10 (10 × 1 = 10, subtract from 11)
----
00
00 (10 × 0 = 00, subtract from 00)
----
 0
```

SHORT QUESTIONS

Q.16 How do you add binary numbers?

Ans: **ADDING BINARY NUMBERS**

Binary addition uses the following rules:
0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 (with carry of 1)
Example: 1101 + 1011 = 11000 (Carry operations are shown for each bit addition).

Q. 2 What is the process of subtracting binary numbers?

Ans: **SUBTRACTING BINARY NUMBERS**

To subtract binary numbers, first find the two's complement of the subtrahend, then add it to the minuend.
Example: 9 - 6 = 3, in binary:
Minuend: 1001
Subtrahend: 0110
Two's complement of 0110: 1001
Add: 1001 + 1010 = 10011
Discard the carry: 0011 (which is 3 in decimal).

Q. 3 Explain binary multiplication with an example.

Ans: **BINARY MULTIPLICATION**

Multiply each bit of the second number by each bit of the first number and shift the partial results to the left.
Example: 101 x 11:
Multiply 101 by 1 (101)
Multiply 101 by the other 1 (shifted left, 1010)
Add the results: 101 + 1010 = 1101.

Q. 4 What are the steps for binary division?

Ans: **STEPS FOR BINARY DIVISION**

Binary division follows these steps:
1. Compare the divisor with the dividend portion.
2. Subtract the divisor from the dividend if possible.
3. Shift the next digit from the dividend down.
4. Repeat until all digits of the dividend are used.
Example: 1100 ÷ 10:
Compare: 10 with 11, subtract: 10 (remainder 0)
Bring down the next digit: 0
Compare: 10 with 10, subtract: 10 (remainder 0)
Result: 110.

Q. 5 How do you perform addition of negative binary numbers?

Ans: **ADDITION OF NEGATIVE BINARY NUMBERS**

Use the two's complement of the negative number and add it to the positive number.
Example: 1101 + (-1011) = 11000 (Carry operations included for the negative number).

Q. 6 What is the significance of the carry bit in binary addition?

Ans: **SIGNIFICANCE OF THE CARRY BIT IN BINARY ADDITION**

The carry bit is used when the sum of two binary digits exceeds 1. It is passed to the next higher bit.
Example: In 1101 + 1011, carry bits are generated in the process.

Q. 7 What is a two's complement and how is it used in binary subtraction?

Ans: **COMPLEMENT AND IT'S USE IN BINARY SUBTRACTION**

The two's complement is obtained by inverting all bits of a binary number and adding 1 to the result. This method is used to subtract binary numbers.

Example: For 6 - 9, the two's complement of 9 (0110) is 1001, and when added to 6 (1001), the result is 0011.

Q. 8 Explain how to multiply binary numbers step-by-step.

Ans: **<u>MULTIPLIPLICATION OF BINARY NUMBERS STEP-BY-STEP</u>**

Binary multiplication involves writing down the binary numbers and multiplying bit by bit, shifting partial results.

Example: 101 x 11:
101 x 1 = 101
101 x 1 (shifted) = 1010
Add: 101 + 1010 = 1101.

Q. 9 What are the rules for binary addition?

Ans: **<u>RULES FOR BINARY ADDITION</u>**

The basic rules for binary addition are:
0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 with carry of 1.
Example: 1101 + 1011 = 11000.

Q. 10 How does binary division resemble decimal division?

Ans: **BINARY AND <u>DECIMAL DIVISION</u>**

Binary division follows the same steps as decimal division: compare, subtract, shift, and repeat.
Example: 1100 ÷ 10:
Compare 10 with 11, subtract, bring down 0, continue.

Q. 11 Explain how the carry bit is used in binary addition.

Ans: **<u>CARRY BIT</u>**

The carry bit is used when the sum of two digits is greater than or equal to 2, and it is transferred to the next higher bit position.
Example: 1 + 1 = 0 with carry 1 to the next higher bit.

Q. 12 What does the binary multiplication process involve?

Ans: **<u>BINARY MULTIPLICATION PROCESS</u>**

The binary multiplication process involves multiplying each bit of one number by each bit of another, then shifting the results to the left.
Example: 101 x 11: Multiply, shift, and add results.

Q. 13 How do you subtract binary numbers with two's complement?

Ans: **<u>BINARY NUMBERS WITH TWO'S COMPLEMENT</u>**

Subtract by adding the two's complement of the subtrahend to the minuend and discarding the carry.
Example: 9 - 6 = 3 in binary.

Q. 14 What is the importance of binary arithmetic in computers?

Ans: **<u>IMPORTANCE OF BINARY ARITHMETIC IN COMPUTERS</u>**

Binary arithmetic is crucial in computers for performing mathematical operations efficiently using binary digits (0 and 1).
Example: CPUs use binary arithmetic to perform calculations for tasks like addition, subtraction, multiplication, and division.

Q. 15 Describe the long division method in binary division.

Ans: **<u>LONG DIVISION METHOD IN BINARY DIVISION</u>**

Long division in binary is performed by comparing, subtracting, shifting, and repeating, just like decimal division.
Example: 1100 ÷ 10: Compare, subtract, bring down digits, and repeat until all digits are used.

MULTIPLE CHOICE QUESTIONS

11. What is the result of adding 1101 and 1011 in binary?
(A) 10110 (B) 11000
(C) 11100 (D) 11110

12. Which of the following binary addition rules is correct?
(A) 1+1=1 (B) 1+0=0
(C) 0+0=0 (D) 1+1=0 (with carry of 1)

13. What is the two's complement of 0110?
(A) 1001 (B) 1010
(C) 0111 (D) 1101

14. What is the binary result of subtracting 6 from 9?
(A) 110 (B) 101
(C) 011 (D) 111

15. What is the first step in multiplying two binary numbers?
(A) Add all the partial results
(B) Write down the binary numbers aligned by the least significant bit
(C) Shift the partial results
(D) Compare the numbers

16. Which operation in binary arithmetic follows the same principles as long division in decimal?
(A) Addition (B) Subtraction
(C) Multiplication (D) Division

17. In binary subtraction, how is the subtrahend represented?
(A) By adding the original number
(B) By taking the two's complement of the subtrahend
(C) By shifting the subtrahend
(D) By multiplying the minuend

18. What is the final result of the binary division 1100 ÷ 10?
(A) 100 (B) 101
(C) 110 (D) 111

19. What is the binary multiplication result of 101 x 11?
(A) 101 (B) 1001
(C) 1111 (D) 1101

20. What is the range of exponents for binary addition?
(A) -1 to +1 (B) -126 to +127
(C) 0 to 100 (D) -1022 to +1023

21. How are the binary addition results recorded?
(A) From bottom to top (B) From left to right
(C) From top to bottom (D) From right to left

22. In the process of binary division, when is the next digit of the dividend brought down?
(A) After comparing the divisor (B) After shifting the remainder
(C) After subtracting the divisor (D) After calculating the quotient

23. In binary multiplication, how are partial results shifted?
(A) No shift is performed (B) Shift one place to the right
(C) Shift one place to the left for each new row (D) Shift by two places

24. What is the result of subtracting 9 from 6 in binary?
(A) 0000 (B) 0011
(C) 0101 (D) 1010

25. **Which operation uses the two's complement method in binary arithmetic?**
(A) Addition (B) Subtraction
(C) Multiplication (D) Division

2.5 COMMON TEXT ENCODING SCHEMES

LONG QUESTION

Q. 2 Explain the process of encoding text using ASCII, Extended ASCII, and Unicode with examples.

Ans:

- **ASCII Encoding**:

  ASCII assigns a numerical value between 0 and 127 to each character. It represents basic English characters, digits, and symbols. For example:

  The letter 'P' is encoded as 80 in ASCII.

  The letter 'a' is encoded as 97.

  ASCII is primarily used for basic text communication in early computers.

- **Extended ASCII**:

  Extended ASCII extends the original ASCII by using 8 bits (1 byte) instead of 7 bits, allowing it to represent 256 characters. This extension includes additional symbols, accented letters, and characters from other languages, like 'ç' (code 135).

  For example, 'ç' is represented as 135 in Extended ASCII, which isn't available in standard ASCII.

- **Unicode Encoding**:

  Unicode is a much larger character set designed to accommodate all the world's writing systems. Unlike ASCII, which only supports 128 characters, Unicode supports over a million characters. It uses various encoding schemes like UTF-8, UTF-16, and UTF-32.

  UTF-8 is backward compatible with ASCII and uses a variable number of bytes per character (1 to 4 bytes). For example, the letter 'A' in UTF-8 is represented as 01000001 (1 byte), and the Urdu letter "ب" is represented as 11011000 10101000 (2 bytes).

  UTF-16 uses 2 bytes for most characters and up to 4 bytes for others. The letter 'A' in UTF-16 is represented as 0000000001000001 (2 bytes).

  UTF-32 uses a fixed length of 4 bytes for all characters, which is simple but space-inefficient for many common characters. The letter 'A' in UTF-32 is represented by 00000000 00000000 0000000001000001 (4 bytes).

  Example: Unicode allows text like "Hello" (English) and "ب" (Urdu) to be encoded consistently, ensuring compatibility across different systems and languages

Character	ASCII Code	Character	ASCII Code
SP (space)	32	!	33
"	34	#	35
$	36	%	37
&	38	'	39
(	40	)	41
*	42	+	43
,	44	-	45
.	46	/	47
0	48	1	49
2	50	3	51
4	52	5	53
6	54	7	55
8	56	9	57
:	58	;	59
<	60	=	61
>	62	?	63
@	64	A	65
B	66	C	67
D	68	E	69
F	70	G	71
H	72	I	73
J	74	K	75
L	76	M	77
N	78	O	79

P	80	Q	81
R	82	S	83
T	84	U	85
V	86	W	87
X	88	Y	89
Z	90	[	91
\	92	]	93
^	94	-	95
?	96	a	97
b	98	c	99
d	100	e	101
f	102	g	103
h	104	i	105
j	106	k	107
l	108	m	109
n	110	o	111
p	112	q	113
r	114	s	115
t	116	u	117
v	118	w	119
x	120	y	121
z	122	{	123
|	124	}	125
~	126	DEL	127

SHORT QUESTIONS

Q.2 What is ASCII and how is it used?

Ans:

ASCII AND IT'S USES

ASCII (American Standard Code for Information Interchange) is a character encoding standard used to represent text in computers. It assigns a code number between 0 and 127 to each letter, number, or symbol, allowing devices to exchange text reliably.

Example: The ASCII code for 'P' is 80, and for 'a' is 97.

Q.3 Explain the difference between ASCII and Extended ASCII.

Ans:

DIFFERENCE BETWEEN ASCII AND EXTENDED ASCII

ASCII uses 7 bits to represent 128 characters, while Extended ASCII uses 8 bits, allowing it to represent 256 characters. Extended ASCII includes additional symbols and accented letters.

Example: Extended ASCII includes characters like 'ç' (code 135) which are not present in standard ASCII.

Q.4 What is Unicode, and how does it differ from ASCII?

Ans:

UNICODE, AND HOW DOES IT DIFFER FROM ASCII

Unicode is a character encoding standard that aims to represent all characters from the world's writing systems, unlike ASCII, which can only represent 128 characters. Unicode uses multiple encoding schemes like UTF-8, UTF-16, and UTF-32.

Example: The letter 'A' in Unicode is represented as U+0041, and Unicode can represent over a million characters.

Q.5 How does UTF-8 work, and why is it compatible with ASCII?

Ans:

UTF-8 WORK, AND WHY IS IT COMPATIBLE WITH ASCII

UTF-8 is a variable-length encoding scheme that can use 1 to 4 bytes to represent a character. It is backward compatible with ASCII because it uses the same encoding for the first 128 characters, ensuring ASCII texts are also valid in UTF-8.

Example: 'A' in UTF-8 is represented as 01000001, the same as in ASCII.

Q.6 What is the maximum byte size required for a character in UTF-8?

Ans:

MAXIMUM BYTE SIZE REQUIRED FOR A CHARACTER IN UTF-8

UTF-8 uses a variable number of bytes, ranging from 1 to 4 bytes per character, depending on the complexity of the character.

Example: The letter 'A' requires 1 byte, while some complex characters like those in non-Latin scripts may require up to 4 bytes.

Q.7 How does UTF-16 encoding work?

Ans:

UTF-16 ENCODING WORK

UTF-16 is a variable-length encoding scheme that uses 2 bytes for most characters but can use up to 4 bytes for certain characters. It is not backward compatible with ASCII.

Example: The letter 'A' is represented as 0000000001000001 (2 bytes) in UTF-16.

Q.8 What is the fixed byte length of UTF-32?

Ans: UTF-32 uses a fixed length of 4 bytes per character. While this simplifies encoding and decoding, it can lead to inefficient space usage for simpler characters.

Example: The letter 'A' in UTF-32 is represented by 00000000 00000000 0000000001000001.

Q.9 Why is Unicode important in modern computing?

Ans:

UNICODE IMPORTANTANCE IN MODERN COMPUTING

Unicode allows computers to represent text from all languages and symbols worldwide, enabling global communication and data exchange. It is essential for supporting multiple languages and scripts.

Example: Unicode allows representing text in languages like Arabic, Chinese, and Hindi without conflicts.

Q.10 What are the advantages of using UTF-8 over UTF-16?

Ans:

ADVANTAGES OF USING UTF-8 OVER UTF-16

UTF-8 is more efficient for texts containing primarily ASCII characters because it uses only 1 byte for these characters, whereas UTF-16 always uses 2 bytes.

Example: A text file containing mostly English text will be smaller in UTF-8 than in UTF-16.

Q.11 How does the encoding of Urdu characters differ in UTF-8 and UTF-16?

Ans: In UTF-8, the Urdu character "ب" is represented as 11011000 10101000, occupying 2 bytes. In UTF-16, the same character is represented with 2 bytes (00000110 00101000).

Example: UTF-8 uses 2 bytes for this character, while UTF-16 uses a similar number of bytes but encodes it differently.

MULTIPLE CHOICE QUESTIONS

21. What does ASCII stand for?

(A) Advanced System for Character Information
(B) American Standard Code for Information Interchange
(C) American System of Computer Information
(D) Advanced System Code for Information

22. How many characters does the standard ASCII code represent?

(A) 64 (B) 128
(C) 256 (D) 512

23. Which encoding system uses 8 bits and extends ASCII to 256 characters?

(A) Unicode (B) Extended ASCII
(C) UTF-8 (D) UTF-16

24. What is the maximum number of characters that Unicode can represent?

(A) 128 (B) 256
(C) Over a million (D) 100,000

25. Which of the following encoding formats is backward compatible with ASCII?

(A) UTF-16 (B) UTF-8
(C) UTF-32 (D) Extended ASCII

26. What does UTF stand for?

(A) Universal Text Format (B) Unicode Transformation Format
(C) Universal Translation Format (D) Unified Text Format

27. How many bytes does UTF-8 use to represent the letter 'A'?

(A) 1 byte (B) 2 bytes
(C) 3 bytes (D) 4 bytes

28. Which encoding format uses a fixed length of 4 bytes per character?

(A) UTF-8 (B) UTF-16
(C) UTF-32 (D) Extended ASCII

29. How many bytes does UTF-16 use to represent the letter 'A'?

(A) 1 byte (B) 2 bytes
(C) 3 bytes (D) 4 bytes

30. What is the binary representation of the Urdu letter "ب" in UTF-8?

(A) 11011000 10101000 (B) 11011000 10101001
(C) 11000000 10101000 (D) 11101000 10101001

2.6 STORING IMAGES, AUDIO AND VIDEO IN COMPUTERS

LONG QUESTION

Q.11 Explain how images, audio, and video are stored in computers, including the role of file formats, data sizes, and storage devices.

Ans: Storing Images: Images are made up of tiny dots called pixels, each having a color represented by three numbers in the RGB format (Red, Green, Blue). Each of these numbers typically ranges from 0 to 255, and the combination of all pixels forms the image.

- **File Formats:** Common image formats include JPEG (compresses images to save space but may lose quality), PNG (maintains high quality without losing data), and GIF (used for animations and images with few colors).
- **Data Size:** Image files can range from a few KBs (Kilobytes) for small images to several MBs (Megabytes) for high-resolution images.
- **File Formats:** MP3 is a compressed format that saves space but may lose some quality, WAV is uncompressed and maintains high quality, and AAC is used by streaming services for efficient compression and good audio quality.
- **Data Size:** Audio files vary in size, with typical music files ranging from 3 MB to 10 MB, depending on the quality and length.
- **Storing Video:** Videos are made up of many images (frames) shown in rapid sequence, along with audio. The frame rate defines how many frames are shown per second, with 24 fps commonly used in movies and 30 fps in TV.
- **File Formats:** MP4 is widely used for efficient compression while maintaining quality, AVI is an older format resulting in larger file sizes, and MKV supports high-quality video and multiple audio tracks or subtitles.
- **Data Size:** Videos are typically stored in GBs (Gigabytes), with a 2 GB file size being common for HD movies.
- **Storage Devices:**

a. **Hard Disk Drive (HDD):** Uses spinning disks to read/write data and offers large storage capacities.
b. **Solid State Drive (SSD):** Uses flash memory for faster access times and better performance.
c. **Cloud Storage:** Stores files on remote servers accessible via the internet, providing flexibility and backup options.

These storage methods and technologies allow us to store and access images, audio, and video efficiently in our daily lives.

SHORT QUESTIONS

Q. 1 What is the role of pixels in storing images?

Ans: Pixels are the tiny dots that make up an image. Each pixel has a color, and when combined, these pixels form the complete image. Colors are represented using numbers, typically in RGB format (Red, Green, Blue).

Example: A pixel with RGB values (255, 0, 0) is bright red.

Q. 2 How do sampling and quantization work in storing audio?

Ans: Sampling involves recording the sound wave at regular intervals, known as the sampling rate. Quantization then converts each of these samples into a number. Higher sampling rates and more bits per sample result in better sound quality.

Example: Higher sampling rates, such as 44.1 kHz, result in better audio quality.

Q. 3 Why are file formats like MP3 used for audio storage?

Ans: MP3 is used because it compresses audio files to save space, which makes sharing and storing them more efficient. However, this compression may result in some loss of quality.

Example: MP3 files are much smaller than WAV files, allowing faster downloads and streaming.

Q. 4 What is the importance of frame rate in video storage?

Ans: The frame rate determines how many frames are shown per second in a video. Higher frame rates result in smoother motion. Common frame rates include 24 fps for movies and 30 fps for TV.
Example: A 24 fps video will appear less smooth compared to a 60 fps video.

Q. 5 How does the MP4 video format differ from AVI and MKV?

Ans: MP4 is a widely used video format that efficiently compresses videos while maintaining quality. AVI is an older format that results in larger file sizes, and MKV supports high-quality video and multiple audio tracks or subtitles.
Example: MP4 is ideal for streaming, while AVI is often used for raw footage storage.

Q. 6 What is the difference between HDD and SSD in terms of storage?

Ans: HDDs (Hard Disk Drives) use spinning disks to read and write data, offering large storage capacities. SSDs (Solid State Drives) use flash memory, which provides faster access times and better performance.
Example: SSDs are faster for loading applications, while HDDs are cheaper for large data storage.

Q. 7 Why is Cloud Storage flexible for users?

Ans: Cloud Storage allows users to store files on remote servers accessible via the internet. This offers flexibility as users can access files from any device with an internet connection and provides options for data backup.
Example: Google Drive and Dropbox are popular cloud storage services.

Q. 8 What does RGB stand for in image representation?

Ans: RGB stands for Red, Green, and Blue. It is a color model used in digital imaging where each pixel's color is represented by three numbers corresponding to these primary colors, typically in a range from 0 to 255.
Example: RGB(255, 0, 0) represents a bright red color.

Q. 9 What is the significance of binary data in computer storage?

Ans: All data in computers, including images, audio, and video, is stored as binary data, which is represented as sequences of 0s and 1s. This allows computers to process and store information efficiently.
Example: A text file or image is ultimately converted into binary code for storage.

Q. 10 How do computers handle large files like HD movies?

Ans: Large files, such as HD movies, are stored in binary format and may require large storage capacities, such as Gigabytes or Terabytes. Compression techniques are often used to reduce file size while maintaining quality.
Example: A 2 GB HD movie may be compressed to 1.5 GB using video codecs like MP4.

MULTIPLE CHOICE QUESTIONS

Q. 20 What does 1 Kilobyte (K(B) equal in Bytes?

(A) 512 Bytes (B) 1024 Bytes
(C) 2048 Bytes (D) 1000 Bytes

Q. 21 Which of the following is a commonly used image format that compresses the image but may lose some quality?

(A) PNG (B) GIF
(C) JPEG (D) TIFF

Q. 22 What is the unit used to represent the size of an audio file?

(A) Pixels (B) Bytes
(C) Frames (D) Sampling Rate

Q. 23 Which audio file format is uncompressed and maintains high quality?

(A) MP3	(B) WAV
(C) AAC	(D) FLAC

Q. 24 What is the common frame rate for movies?

(A) 30 fps	(B) 15 fps
(C) 24 fps	(D) 60 fps

Q. 25 Which video file format is known for efficiently compressing video while maintaining quality?

(A) AVI	(B) MKV
(C) MP4	(D) MOV

Q. 26 Which storage device uses flash memory for faster access times and better performance?

(A) HDD	(B) SSD
(C) Cloud Storage	(D) Optical Disk

Q. 27 What is the binary representation of data in computers?

(A) Colors and Sounds	(B) Sequences of 1s and 0s
(C) Digital Frequencies	(D) Audio Samples

Q. 28 What is the typical size of a music file?

(A) 50 KB	(B) 500 KB
(C) 5 MB	(D) 500 MB

Q. 29 What is the advantage of using Cloud Storage for files?

(A) Faster processing speeds	(B) Large storage capacity
(C) Flexibility and backup options	(D) Unlimited storage

SUMMARY

- In computing, numbering systems are crucial as they form the foundation for representing, storing, and processing information.
- Decimal number system is a number system in which base is 10 and the digits involved are 0 to 9, which are commonly used in our daily lives.
- Binary is a base-2 number system that comprises of only the digits 0 and 1. Each digit represents a power of two.
- The Octal number system is another number system that has eight as its base; thus, it has eight digits 0 to7. Each digit represents a power of 8, this can be expressed as 8 digit.
- The Hexadecimal numbering system is another type of number system with base of 16, where the number 0 to 9 and alphabets A-F are used.
- Integers refers to the set of non-negative whole numbers, while whole numbers are the complete numbers. They include zero and all the positive integers, also positive zero.
- To store negative values, computers employ a technique commonly known as two's complement.
- In computers, real values, which are nicknamed as floating-point numbers are used to represent numbers with fraction or decimal point.
- Arithmetic operations mean addition, subtraction multiplication, and division performed on numbers in given base. Binary arithmetic involves performing these operations on numbers in binary form, or base 2.
- ASCII is an acronym for American Standard Code for Information Interchange. It is an industry standard used to encode text in computers and other devices.
- All the letters, digits or symbols are given a unique number ranging from 0 to 127.
- According to the standard ASCII table, there are both a basic version with 128 characters and an extending version with 256 characters.
- Unicode is a character encoding standard that aims to provide character codes for all the characters use a in the worlds' writing systems.

- UTF-16 on the other hand is another variable character encoding technique in which characters are represented using either two or four bytes.
- UTF-32 is a blocking encoding method of fixed block size, meaning that the size of each block is constant. Each single character is always represented using 4 bytes, regardless of its specific characteristics.
- Images under digital context are composed of small points referred to as pixel. Each of them is of certain color and the combination of such pixels makes a complete image or picture.
- Audio files are recorded using the human-computer interface to capture and then converting sound waves into digital form. Some of the key steps in this process include data sampling and quantization.

EXERCISE

MULTIPLE-CHOICE QUESTIONS

1. **What does ASCII stand for?**
(A) American Standard Code for Information Interchange
(B) Advanced Standard Code for Information Interchange
(C) American Standard Communication for Information Interchange
(D) Advanced Standard Communication for Information Interchange

2. **Which of the following numbers is a valid binary number?**
(A) 1101102 (B) 11011
(C) 110.11 (D) 1101A

3. **How many bits are used in the standard ASCII encoding?**
(A) 7 bits (B) 8 bits
(C) 16 bits (D) 32 bits

4. **Which of the following is a key advantage of Unicode over ASCII?**
(A) It uses fewer bits per character
(B) It can represent characters from many different languages
(C) It is backward compatible with binary
(D) It is specific to the English language

5. **How many bytes are used to store a typical integer?**
(A) 1 byte (B) 2 bytes
(C) 4 bytes (D) 8 bytes

6. **What is the primary difference between signed and unsigned integers?**
(A) Unsigned integers cannot be negative
(B) Signed integers have a larger range
(C) Unsigned integers are stored in floating-point format
(D) Signed integers are only used for positive numbers

7. **In the single precision, how many bits are used for the exponent?**
(A) 23 bits (B) 8 bits
(C) 11 bits (D) 52 bits

8. **What is the approximate range of values for single-precision floating-point numbers?**
(A) 1.4 * 10 ^ - 45 * to * 3.4 * 10 ^ 36 (B) 1.4 * 10 ^ - 16 * to * 3.4 * 10 ^ 45
(C) 1.8 * 10 ^ (30w) 4.9 * 10 ^ - 524 * 1 (D) 4.9 * 10 ^ 0.2 to 1.8 * 10 ^ 324

9. **What are the tiny dots that make up an image called?**
(A) Pixels (B) Bits
(C) Bytes (D) Nodes

10. **In an RGB color model, what does RGB stand for?**
(A) Red, Green, Blue (B) Red, Gray, Black
(C) Right, Green, Blue (D) Red, Green, Brown

SHORT QUESTIONS

Q. 1 What is the primary purpose of the ASCII encoding scheme?

Ans: The primary purpose of the ASCII (American Standard Code for Information Interchange) encoding scheme is to represent text characters in computers and communication devices. It assigns a unique numeric value to each character (letters, digits, punctuation, and control characters) to enable standardized data exchange between systems.

Q. 2 Explain the difference between ASCII and Unicode.

Ans:

1. **Character Set:**
   - ASCII represents a limited set of 128 characters (7-bit encoding).
   - Unicode can represent a vast number of characters (over 1.1 million) to support almost all written languages worldwide.
2. **Encoding Size:**
   - ASCII uses 7 or 8 bits per character.
   - Unicode uses variable encoding formats like UTF-8 (1-4 bytes), UTF-16 (2 or 4 bytes), or UTF-32 (4 bytes).
3. **Language Support:**
   - ASCII is limited to the English alphabet and a few symbols.
   - Unicode supports characters from many languages, scripts, and special symbols.

Q. 3 How does Unicode handle characters from different languages?

Ans: Unicode assigns a unique code point to each character, regardless of language. These code points are represented in hexadecimal notation (e.g., U+0041 for 'A'). Encoding formats like UTF-8 or UTF-16 allow efficient storage and transmission of these code points while maintaining compatibility with older systems.

Q. 4 What is the range of values for an unsigned 2-byte integer?

Ans: A 2-byte integer consists of 16 bits. For an **unsigned integer** (only positive values):

$$\text{Range}=0 \text{ to } (2^{16}-1)=0 \text{ to } 65{,}535$$

Q. 5 Explain how a negative integer is represented in binary.

Ans: Negative integers are represented in binary using the **two's complement** method:

1. Convert the absolute value of the number to binary.
2. Invert all bits (ones' complement).
3. Add 1 to the inverted result.

For example to represent −5 in an 8-bit system:

1. Absolute value of 5 = 00000101
2. Invert bits = 11111010
3. Add 1 = 11111011

Q. 6 What is the benefit of using unsigned integers?

Ans: **Larger Range:**

Unsigned integers can represent a larger range of positive values because they do not need to allocate a bit for the sign.

Non-Negative Quantities:

They are ideal for scenarios where negative values are meaningless, such as counters, indexes, and memory addresses.

Q. 7 How does the number of bits affect the range of integer values?

Ans: The range of integer values depends on the number of bits:

1. **Unsigned Integer:** Range=0 to $(2^{n}-1)$
2. **Signed Integer (Two's Complement):** Range=$-2^{(n-1)}$ to $(2^{(n-1)}-1)$.

Increasing the number of bits increases the range exponentially.

Q. 8 Why are whole numbers commonly used in computing for quantities that cannot be negative?

Ans: Whole numbers are used in such cases because:

- They simplify arithmetic and logical operations by avoiding negative values.
- They efficiently utilize memory (e.g., unsigned integers have a larger range for positive numbers).
- Examples include quantities like population counts, array indexes, and timestamps.

Q. 9 How is the range of floating-point numbers calculated for single precision?

Ans: In single-precision floating-point representation (32 bits):

1. **Structure:**
- 1 bit for the sign (positive/negative).
- 8 bits for the exponent (with bias = 127).
- 23 bits for the fraction (mantissa).
2. **Range:**
- Smallest Positive Value: 2^{-126}
- Largest Positive Value: $(2-2^{-23})\times2^{127}$

The actual range depends on the precision and exponent bias.

3. **Why is it important to understand the limitations of floating-point representation in scientific computing?**

1. **Precision Errors:**
Floating-point numbers cannot represent all real numbers exactly, leading to rounding errors.
2. **Loss of Accuracy:**
Repeated calculations can accumulate small errors, causing significant deviations in results.
3. **Overflow and Underflow:**
Extremely large or small numbers may exceed the representable range.
4. **Comparison Issues:**
Due to precision limitations, equality checks can fail even for seemingly identical values.

Understanding these limitations helps design robust algorithms and ensures numerical stability in computations.

LONG QUESTIONS

1. **Explain how characters are encoded using Unicode. Provide examples of characters from different languages and their corresponding Unicode code points.**

1. Overview of Unicode Encoding

Unicode is a universal character encoding standard that assigns a unique code point to characters from virtually all the world's writing systems. It ensures consistent representation, storage, and transmission of text across different systems and platforms. Unlike ASCII, which is limited to 128 characters, Unicode can represent over a million characters, covering alphabets, symbols, and emojis.

2. How Unicode Encodes Characters

Unicode assigns each character a unique **code point** represented in the format **U+XXXX**, where "XXXX" is a hexadecimal number. The encoding is implemented using various forms, such as **UTF-8**, **UTF-16**, and **UTF-32**.

- **UTF-8:**
  - A variable-length encoding that uses 1 to 4 bytes per character.
  - It is backward-compatible with ASCII, meaning ASCII characters retain their original encoding.
  - Example: The letter "A" is encoded as **U+0041** and stored in binary as **01000001** (1 byte).
- **UTF-16:**
  - Uses 2 or 4 bytes for each character.
  - It is not compatible with ASCII.
  - Example: The letter "A" is represented as **00000000 01000001** in binary (2 bytes).

- **UTF-32:**
  - A fixed-length encoding where each character is stored in 4 bytes.
  - Example: The letter "A" is encoded as **00000000 00000000 00000000 01000001** (4 bytes).

3. Examples of Unicode Characters

a. English Letters

- **Character:** "A"
  - **Code Point:** U+0041
  - **UTF-8 Binary Representation:** 01000001

b. Urdu Letters

- **Character:** "ب"
  - **Code Point:** U+0628
  - **UTF-8 Binary Representation: 11011000 10101000** (2 bytes)
  - **UTF-16 Binary Representation: 00000110 00101000**

c. Japanese Characters

- **Character:** "日" (meaning "sun")
  - **Code Point:** U+65E5
  - **UTF-8 Binary Representation: 11100110 10011110 10010101** (3 bytes)

d. Emoji

- **Character:** "😊"
  - **Code Point:** U+1F60A
  - **UTF-8 Binary Representation: 11110000 10011111 10011000 10101010** (4 bytes)

4. Advantages of Unicode

- **Global Coverage:** Supports characters from virtually all languages and scripts.
- **Compatibility:** UTF-8 is backward-compatible with ASCII, making it versatile for older systems.
- **Scalability:** With more than a million code points, Unicode accommodates modern needs like emojis and special symbols.

2. **Describe in detail how integers are stored in computer memory.**

1. Overview of Integer Storage in Computers

Integers, or whole numbers, are fundamental data types in computer systems. They are stored in memory as binary values (a series of 0s and 1s). Depending on whether they are **unsigned integers** (non-negative) or **signed integers** (include both positive and negative values), different encoding methods are used to store them.

2. Storage of Unsigned Integers

Unsigned integers represent non-negative whole numbers, starting from 0. They use all available bits to store the magnitude of the number.

- **Storage Representation:**
  - A 1-byte integer uses 8 bits, allowing for values ranging from 0 to 2^{8-1} (0 to 255).
  - A 2-byte integer uses 16 bits, with a range of 0 to 2^{16-1} (0 to 65,535).
  - A 4-byte integer uses 32 bits, supporting values from 0 to 2^{32-1} (0 to 4,294,967,295).
- **Example (1-byte):**
  - The number 5 is represented as **00000101** in binary.

3. Storage of Signed Integers

Signed integers can store both positive and negative values. The **most significant bit (MSB)** is reserved as the **sign bit**:

- **0** indicates a positive value.
- **1** indicates a negative value.

a. Range of Values:

- A 1-byte signed integer (8 bits) has a range of -2^{7}-2^{7-1} (-128 to 127).
- A 2-byte signed integer (16 bits) has a range of -2^{15}-2^{15-1} (-32,768 to 32,767).
- A 4-byte signed integer (32 bits) has a range of -2^{31}-2^{31-1} (-2,147,483,648 to 2,147,483,647).

b. Negative Value Storage: Two's Complement Method

Negative integers are stored using **two's complement**, which simplifies arithmetic operations.

- **Steps to Compute Two's Complement:**

1. Write the binary representation of the positive value.
2. Invert all the bits (change 0s to 1s and 1s to 0s).
3. Add 1 to the least significant bit (LSB).

- **Example:** Storing -5 in an 8-bit signed integer:

1. Binary of 5: **00000101**
2. Invert: **11111010**
3. Add 1: **11111011**
4. Thus, -5 is stored as **11111011**.

4. Key Characteristics of Integer Storage

a. Signed vs. Unsigned Integers:

- Signed integers allocate one bit for the sign, reducing the range of magnitudes.
- Unsigned integers use all bits for magnitude, providing a larger positive range.

b. Memory Size Impact:

- Larger memory sizes (e.g., 2 or 4 bytes) allow storage of larger integer values.

c. Overflow and Underflow:

- **Overflow** occurs when a value exceeds the maximum representable value (e.g., storing 130 in an 8-bit signed integer).
- **Underflow** occurs when a value is smaller than the minimum representable value (e.g., trying to store -140 in an 8-bit signed integer).

5. Importance of Integer Storage

Efficient integer storage is essential for various computing tasks, including arithmetic operations, data manipulation, and memory management. The use of two's complement for signed integers enables seamless integration of positive and negative values in calculations.

3. **Explain the process of converting a decimal integer to its binary representation and vice versa. Include examples of both positive and negative integers.**

See Topic 2.2

4. **Perform the following binary arithmetic operations:**

a. Multiplication of 101 by 11.

b. Division of 1100 by 10.

a. Multiplication of 101 **by** 11**:**

```
    101
×   11
----------
    101   (101 × 1)
+  1010  (101 × 1, shifted one place left)
----------
   1111   (result)
```

Ans:

101×11=1111

b. Division of **1100** by **10**:

1. **Write the numbers in binary:**
   Dividend: 1100
   Divisor: 10
2. **Perform binary long division:**

```
110 (quotient)
-----
10 | 1100 (dividend)
-10
----
100
-10
----
00
```

Q. 1 Add the following binary numbers

a. **101+110**

1. **Align the binary numbers and add bit by bit, starting from the right:**

```
 101
+ 110
-----
 1011
```

b. **1100+1011**

2. Align the numbers and add bit by bit with carry:

```
  1100
+ 1011
------
 10111
```

Q. 2 Convert the following numbers to 4-bit binary and add them

(a) 7 + (-4):

Step 1: Represent 7 in 4-bit binary

- 7 in decimal is 0111 in 4-bit binary.

Step 2: Represent -4 in 4-bit binary (using two's complement)

1. Start with 4 in binary: 0100.
2. Invert the bits: 1011.
3. Add 1 to the result: 1011+1=1011 + 1 = 1011+1=1100.

- So, -4 is represented as 1100.

Step 3: Add the two numbers

0111 (+7)+1100 (−4)=0011 (+3)0111

Final Answer for (a):

The result of $7+(-4)$ is **3**, represented in binary as 0011.

(b) -5 + 3:

Step 1: Represent -5 in 4-bit binary (using two's complement)

1. Start with 5 in binary: 0101.
2. Invert the bits: 1010.
3. Add 1 to the result: 1010+1=1010 + 1 = 1010+1=1011.

- So, -5 is represented as 1011.

Step 2: Represent 3 in 4-bit binary

- 3 in decimal is 0011 in binary.

Step 3: Add the two numbers

1011 (−5)+0011 (+3)=1110 (−2)

Step 4: Verify the result

- 1110 is the two's complement representation of -2:
    1. Invert 1110 to 0001.
    2. Add 1: 0001+1=0001 + 1 = 0001+1=0010.
    3. So, 1110 represents −2-2−2.

Final Answer for (b):
The result of −5+3 is **-2**, represented in binary as 1110.
Summary of Results:

- **(a)** 7+(−4)= 3, Binary: 0011.
- **(b)** −5+3=−2, Binary: 1110.

Q. 3 Solve the following

a. **$1101_2 - 0100_2$**:

1. **Take the two's complement of** 0100_2 **:**
    - Invert bits: 0100→1011
    - Add 1: 1011+1=1100.
2. **Perform binary addition:**

```
  1101
+ 1100
------
 11001
```

Discard the leftmost bit (5th bit in this case).
Result:
1101−0100=1001

b. $1010_2 - 0011_2$

1. **Take the two's complement of 0011_2:**
    - Invert bits: 0011→1100
    - Add 1: 1100+1=1101.
2. **Perform binary addition:**

```
  1010
+ 1101
------
 10111
```

Discard the leftmost bit.
Result:
1010−0011=0111

c. $1000_2 - 0110_2$:

1. **Take the two's complement of 0110_2:**
    - Invert bits: 0110→1001
    - Add 1: 1001+1=1010.
2. **Perform binary addition:**

```
  1000
```

```
+ 1010
------
 10010
```

Discard the leftmost bit.

Result:

1000−0110=0010

d. 1110_2-100_2:

1. Convert 100_2 to a 4-bit number: 0100_2.
2. Take the two's complement of 0100_2:
    - Invert bits: 0100→1011
    - Add 1: 1011+1=1100.
3. Perform binary addition:

```
  1110
+ 1100
------
 11010
```

Discard the leftmost bit.

Result:

1110−100=1010

ANSWER KEY

TOPIC 2.1 NUMBER SYSTEMS

1	B	2	C	3	C	4	A	5	B
6	C	7	D	8	C	9	A	10	C
11	C	12	A	13	B	14	A	15	A
16	A	17	B	18	C	19	B	20	A
21	A	22	C	23	D	24	B	25	A
26	D	27	A	28	B	29	C	30	D

TOPIC 2.2 DATA REPRESENTATION IN COMPUTING SYSTEMS

1	B	2	B	3	A	4	B	5	B
6	A	7	A	8	B	9	B	10	C
11	B	12	B	13	B	14	A	15	B

TOPIC 2.3 STORING REAL VALUES IN COMPUTER MEMORY

1	B	2	A	3	C	4	A	5	B
6	B	7	C	8	C	9	B	10	D
11	C	12	C	13	B	14	B	15	C

TOPIC 2.4 BINARY ARITHMETIC OPERATIONS

1	B	2	D	3	A	4	C	5	B
6	D	7	B	8	C	9	D	10	B
11	C	12	C	13	C	14	A	15	B

TOPIC 2.5 COMMON TEXT ENCODING SCHEMES

1	B	2	B	3	B	4	C	5	B

TOPIC 2.6 STROING IMAGES, AUDIO AND VIDEO IN COMPUTERS

1	B	2	C	3	B	4	B	5	C
6	C	7	B	8	B	9	C	10	C

TEXTBOOK EXERCISE MCQs

1	A	2	B	3	A	4	B	5	C
6	A	7	B	8	A	9	A	10	A