Unit 03

Data Representation

Introduction

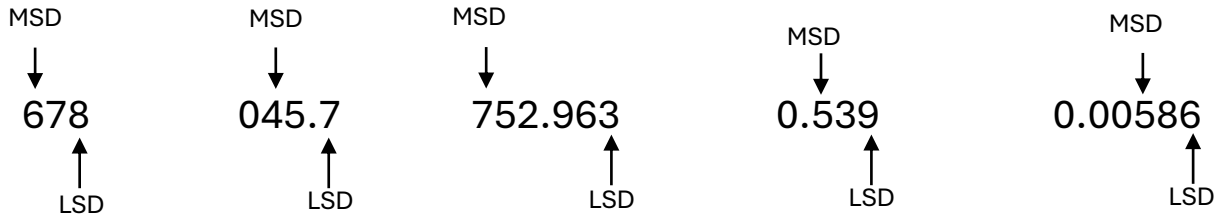
When typing letters or words on a computer, these characters are represented by the computer as numbers it can understand. This group of numbers is called a 'Number System', and the limited set of symbols in this system is known as 'digits'. Computers ultimately represent and process everything as bits. These binary levels are referred to as the high voltage level and the low voltage level. The digits '1' and '0' respectively represent these high and low voltage levels in a circuit. Thus, the '1' and '0' states are equivalent to the "On" and "Off" states of an electronic circuit. Any data in the world can be represented on a computer using these two digits.

In addition to binary, computers also use octal and hexadecimal number systems. These systems simplify binary representation, with octal using base-8 and hexadecimal using base-16.

Most and Least Significant Positional Value of a Number

There are two distinct methods to determine the most and least significant values of whole numbers and decimal numbers. For whole numbers, read from left to right, the rightmost digit is the least significant value, and the leftmost digit that is not zero is the most significant value. In decimal numbers, the rightmost digit after the decimal point that is not zero is the least significant value, and the leftmost digit before the decimal point that is not zero is the most significant value.

Most Significant Digit (MSD) and Least Significant Digit (LSD)



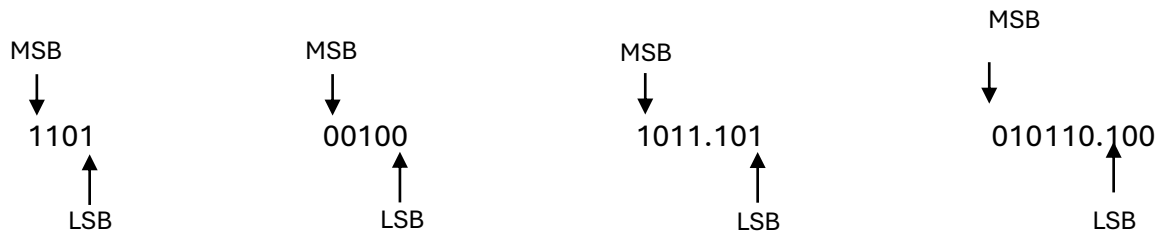
* You can use the same method to find the most and least significant digits of octal, and hexadecimal numbers.

Ex. 01 – Mark the MSD & LSD in the following Decimal numbers.

1) 4057 2) 0127 3) 99.99 4) 0.321 5) 04.065 6) 0.00317

Most Significant Bit (MSB) and Least Significant Bit (LSB)

Only binary number system is used to identify the most significant bit (MSB) and the least significant bit (LSB). For whole numbers, read from left to right, the rightmost bit is the LSB, and the leftmost bit that is not zero is the MSB. For binary decimal numbers, the rightmost bit after the decimal point that is not zero is the MSB, and the leftmost bit before the decimal point that is not zero is the MSB.



Ex 02 – Mark the MSB & LSB in the following Binary numbers.

1) 1011 2) 0101010 3) 1100.011 4) 1110.1110 5) 001100.001100

Decimal Number System

- These are the numbers that we use in day-to-day life.
- The base is 10.
- Digits are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Positive values are represented by (+) and negative values are represented by (-)
- Contains integer values and fractional values. Fractional part is separated by a dot (.)
  Eg - 345.632

Binary Number System

- This system has two digits which can represent two states.
- These two states are represented by digits 1 and 0.
- In binary numbers, place values are multiples of 2.

E.g :

2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}
16	8	4	2	1	.5	.25	.125

➔ **Binary to Decimal**

e.g – 110011_2

1	1	0	0	1	1
↓	↓	↓	↓	↓	↓
2^5	2^4	2^3	2^2	2^1	2^0
↓	↓	↓	↓	↓	↓
32	+ 16	+ ~~8~~	+ ~~4~~	+ 2	+ 1

= 51

101010.11_2

1	0	1	0	1	0	.	1	1
↓	↓	↓	↓	↓	↓		↓	↓
2^5	2^4	2^3	2^2	2^1	2^0		2^{-1}	2^{-2}
↓	↓	↓	↓	↓	↓		↓	↓
32	+ ~~16~~	+ 8	+ ~~4~~	+ 2	+ ~~1~~		+ 1/2	+ 1/4

=42.75

Ex 03 – Convert the following binary numbers into decimal numbers.

1) 111111_2

2) 100010001.011_2

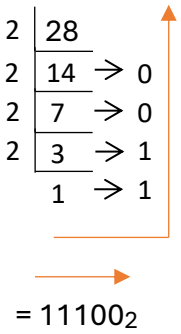
3) 0001010.001_2

4) 1110111.111_2

2^{-4} - 0.0625
2^{-3}- 0.125
2^{-2}- 0.25
2^{-1}- 0.5
2^0- 1
2^1- 2
2^2- 4
2^3- 8
2^4- 16
2^5- 32
2^6- 64
2^7- 128
2^8- 256
2^9- 512
2^{10}- 1024

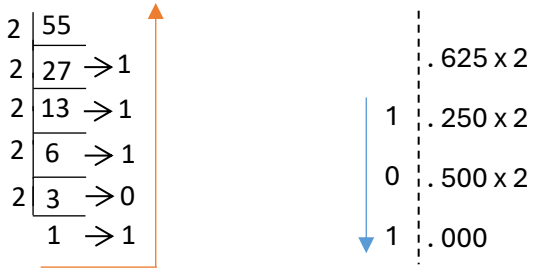
➔ **Decimal to Binary**

Eg - 28



$= 11100_2$

55.625



$= 110111.101_2$

Ex 04 – Convert the following decimal numbers to binary numbers.

1) 31

2) 65

3) 111.75

4) 225.625

5) 710.25

6) 1023.125

Ex 05 – **Binary Addition**

1) $1011_2 + 01110_2$

2) $101.1_2 + 1011.101_2$

3) $11.0_2 + 111.1_2 + 1111.11_2$

4) $101_2 + 1.01_2 + 0.11_2$

5) $1111.011_2 + 101.011_2$

6) $11111.11_2 + 11111.11_2$

Ex 06 – **Binary Subtraction**

1) $1101_2 - 111_2$

2) $10001_2 - 11.11_2$

3) $10000_2 - 1101.01_2 - 101.1_2$

4) $1010.01_2 - 11.11_2$

5) $1111_2 - 0.01_2 - 111.111_2$

6) $11111.1_2 - 1111.001_2 - 11.01_2$

Octal Number System

- Base value is 8.
- Digits used in this representation are 0,1,2,3,4,5,6 and 7.
- In octal numbers, place values are multiples of 8.

E.g :

8^2	8^1	8^0	8^{-1}	8^{-2}
64	8	1	1/8	1/64

➔ Octal to Decimal

e.g – 124_8

1 2 4

↓ ↓ ↓

8^2 8^1 8^0

↓ ↓ ↓

64 + 16 + 4

= 84

166.1_8

1 6 6 . 1

↓ ↓ ↓ ↓

8^2 8^1 8^0 8^{-1}

↓ ↓ ↓ ↓

32 + 16 + 6 + 1/8

=54.125

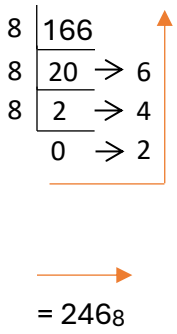
8^{-1}- 0.125
8^0- 1
8^1- 8
8^2- 64
8^3- 512

Ex 07 – Convert the following Octal numbers into Decimal numbers.

1) 153_8 2) 253_8 3) 1364.2_8

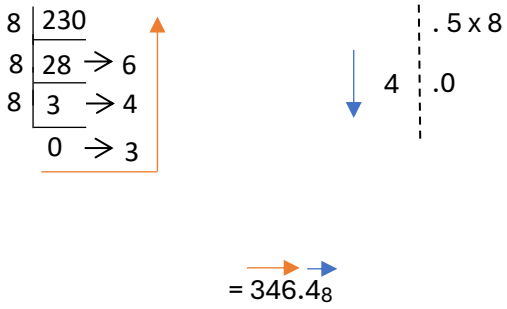
➔ **Decimal to Octal**

Eg - 166



$= 246_8$

230.5



$= 346.4_8$

Ex 08 – Convert the following Decimal numbers into Octal numbers.

1) 512 2) 648.2 3) 1364.4

Ex 09 – **Octal Addition**

1) $153_8 + 316_8$ 2) $145.7_8 + 267_8$ 3) $45.6 + 52.5_8$

4) $777.7_8 + 0.1_8$ 5) $667.62_8 + 112.27_8$ 6) $1001.101_8 + 1100.11_8$

Ex 10 – **Octal Subtraction**

1) $432_8 - 216_8$ 2) $162.3_8 + 24.5_8$ 3) $1000_8 – 1.1_8$

Hexadecimal Number System

- Base value is 16.
- Contains 16 digits starting from 0 to last digit F.
- The values of the digits are represented as follows.

Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

- In octal numbers, place values are multiples of 8.

E.g :

16^2	16^1	16^0	16^{-1}	16^{-2}
256	16	1	1/16	1/256

➔ **Hexadecimal to Decimal**

e.g – $12B_{16}$

1 2 B

↓ ↓ ↓

16^2 16^1 16^0

↓ ↓ ↓

256 + 32 + 11

= 299

$2A1.2_{16}$

2 A 1 . 2

↓ ↓ ↓ ↓

16^2 16^1 16^0 16^{-1}

↓ ↓ ↓ ↓

512 + 160 + 1 + 0.125

=673.125

16^{-1} - 0.0625
16^0 - 1
16^1 - 16
16^2 - 256

Ex 11 – Convert the following hexa-decimal numbers into Decimal numbers.

1) FF_{16} 2) $2C7.4_{16}$ 3) $3DF.8_{16}$

➔ **Decimal to Hexadecimal**

Eg - 999

16 | 999
16 | 62 → 7
16 | 3 → 14
0 → 3

= $3E7_{16}$

422.25

8 | 422
8 | 26 → 6
8 | 1 → 10
0 → 1

. 25 x 16
4 .0

= $1A6.4_{16}$

Ex 12 – Convert the following Decimal numbers into Hexadecimal numbers.

1) 278 2) 512.5 3) 1364.125

Ex 13 – **Hexadecimal Addition**

1) $35A_{16} + 163_{16}$ 2) $69_{16} + 7A_{16}$ 3) $AB.C_{16} + A8.5_{16}$

4) $10_{16} + 25.A_{16} + F.F_{16}$ 5) $53.4_{16} + 55.5_{16} + 99.D_{16}$ 6) $1001.101_{16} + 1100.11_{16}$

Ex 14 – **Hexadecimal Subtraction**

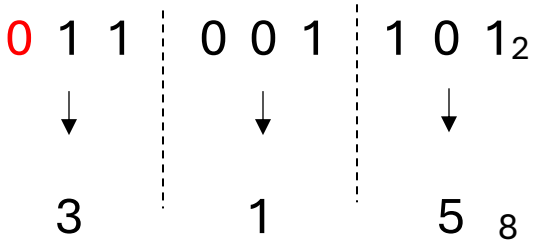
1) $22_{16} - 1A_{16}$ 2) $753B_{16} - A15_{16}$ 3) $38.5_{16} - 19.F_{16}$

Conversions between Number systems

➔ **Binary to Octal**

* Group 3 binary bits each, starting from the right
* Find the total of each group and use the corresponding octal digit.

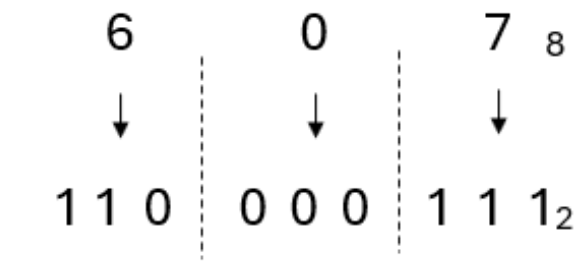
Eg – 11001101_2



➔ **Octal to Binary**

* Represent each octal digit as 3 binary bits

Eg - 607_8



Ex 15 – Convert the following Binary numbers into Octal numbers.

1) 110101001100_2 2) 1111111.01_2 3) 1111.010101_2

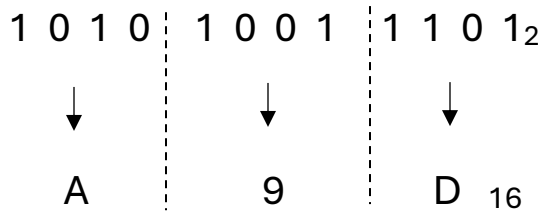
Ex 16 – Convert the following Octal numbers into Binary numbers.

1) 623_8 2) 275.71_8 3) 613.4_8

➔ **Binary to Hexadecimal**

* Group 4 binary bits each, starting from the right
* Find the total of each group and use the corresponding octal digit.

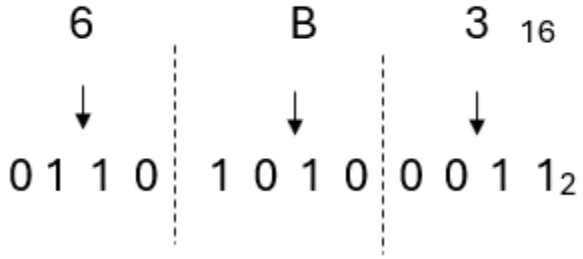
Eg – 101010011101_2



➔ **Hexadecimal to Binary**

* Represent each octal digit as 4 binary bits

Eg – $6B3_{16}$



Ex 17 – Convert the following Binary numbers into Hexadecimal numbers.

1) 1111101110010010_2 2) 110001100.01_2 3) 1101011.11001_2

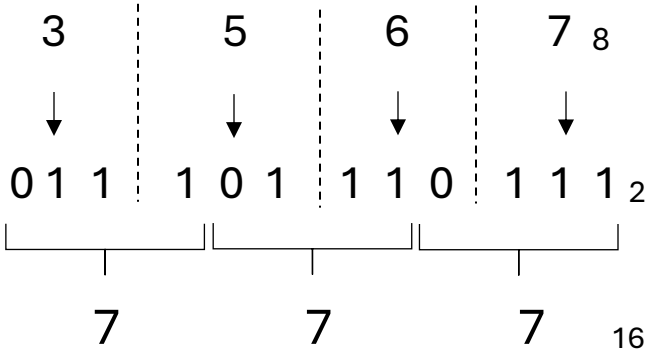
Ex 18 – Convert the following Hexadecimal numbers into Binary numbers.

1) $DF4_{16}$ 2) $1C4.11_{16}$ 3) $619.C3_{16}$

➔ **Octal to Hexadecimal Conversion**

* First, convert to binary number and then Convert to Hexadecimal number.

Eg – $6B3_{16}$



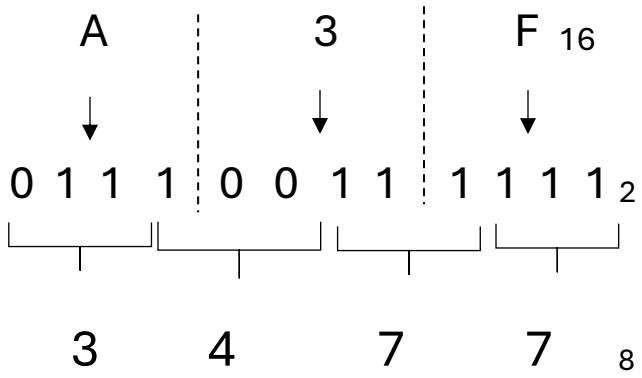
Ex 19 – Convert the following Octal numbers into Hexadecimal numbers.

1) 742_8 2) 164.72_8 3) 557.34_8

➔ **Hexadecimal to Octal Conversion**

* First, convert to binary number and then Convert to Hexadecimal number.

Eg – $6B3_{16}$



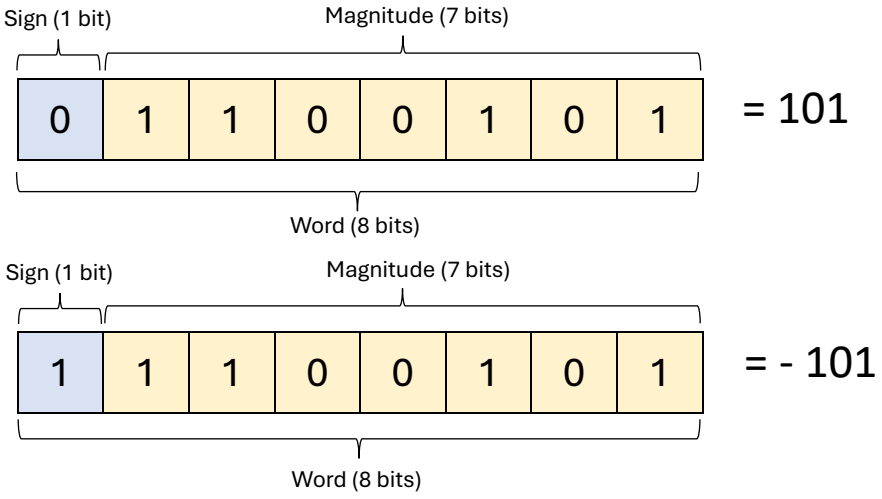
Ex 20 – Convert the following Hexadecimal numbers into Octal numbers.

1) AA_{16} 2) $C1.94_{16}$ 3) $12D.23_{16}$

Number Representation

➔ Sign Magnitude

The sign - magnitude method typically uses an 8-bit system, where the most significant bit (MSB) signifies whether the value is positive or negative. '0' in the MSB denotes that the remaining 7 bits represent a positive number, while a '1' indicates a negative number. This system allows for the representation of a negative zero.



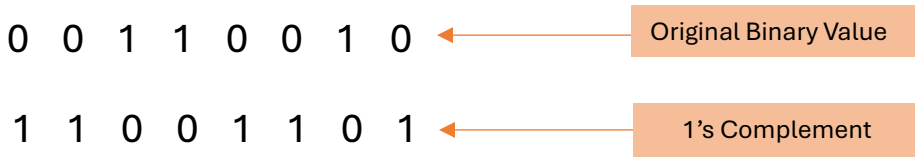
Ex 21 – Convert the following sign-Magnitude numbers to Decimal values

1) 011010110 2) 11001110 3) 10101101

➔ One's Complement

In one's complement notation, positive numbers are represented in standard binary form, while negative numbers are formed by flipping all bits—changing zeros to ones and ones to zeros. For example, 9 is represented as 00001001, and -9 as 11110110. As in sign- magnitude, the most significant bit (MSB) indicates the sign, with 1 for negative and 0 for positive. To find the value of a negative number, you reverse the bits and interpret it as usual.

Eg : -50 in One's complement



The ones complement system has two ways of writing 010 (000000002 = +010 and 111111112 = −010)

Ex 22 – Convert the following numbers to one's complement values

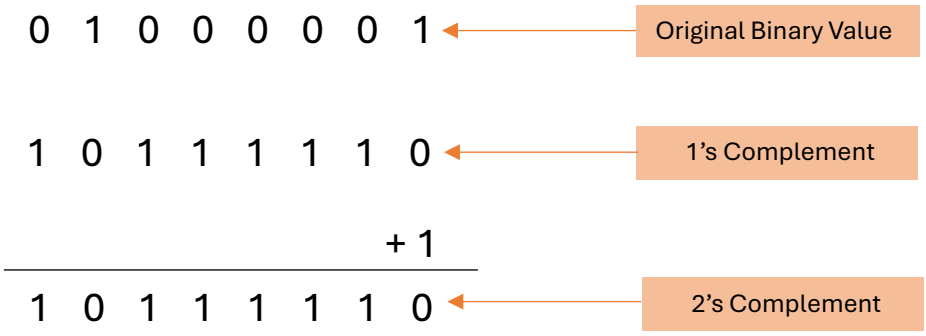
1) 126 2) -127 3) 88

1) -64 2) -52 3) 100

➔ 2's Complement

In two's complement notation, positive numbers are represented same as one's complement, and to obtain the negative value of a number, invert all the bits of its positive value and then add one to the result. This system provides an additional negative value compared to positive values, which helps accommodate zero.

Eg : - 65



Ex 23 – Convert the following numbers to two's complement values

1) 127 2) -127 3) -1

1) -65 2) -55 3) -118

Addition using 1's complement and 2's complement

- Computation 20+(-5) is done in 8-bit two's complement form.

20 → 00010100

5 → 00000101

- 5 → 11111010 + 00000001 = 11111011

```
  00010100
  11111011+
 100001111
```

Carrier bit

*** In one's Complement additions, carrier bit is added again to the answer, But in Two's complement, carrier bit is ignored**

Ex 24 –Compute the following decimal values in two's complement values.

1) 96 + (-64) 2) 127 + (- 40) 3) 28 - 27

Differences between 1's Complement and 2's Complement

1's Complement	2's Complement
1. Zero has two representations, +0 and -0	Zero has only one representation
2. Negative values obtained by flipping bits	To obtain a negative value, should flip bits and add 1
3. More complex for arithmetic operations due to considering carrier bit	Much easier in arithmetic operations
4. Less used in modern computing systems	Widely used in modern computer systems

Fixed point numbers

In calculations involving fixed point numbers that have a fixed number of digits after the decimal point.

Eg :- 850.753
652.365 +
1503.118

* The decimal point is unchanged.

Floating point numbers

Floating-point numbers are used to represent:
- Numbers with fractions, such as 3.1416
- Very small numbers, like 0.000000001
- Very large numbers, for example, 3.15576 × 10^9

There are various ways to represent floating-point numbers, but IEEE 754 is among the most commonly used due to its efficiency. IEEE 754 standard has three main components:

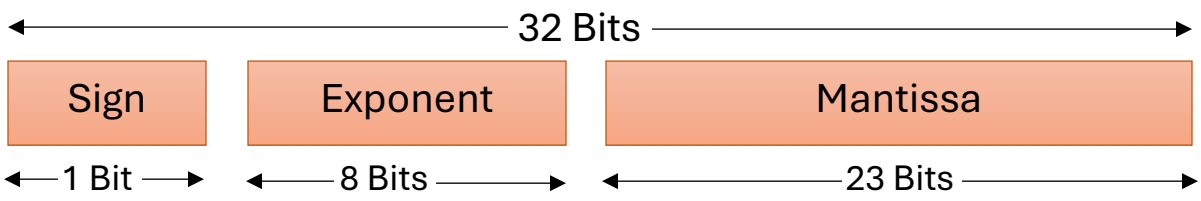
1. **Sign Bit**: This indicates the sign of the number. A sign bit of 0 represents a positive number, while a sign bit of 1 represents a negative number.

2. **Exponent**: The exponent field needs to represent both positive and negative exponents. A bias is added to the actual exponent in order to get the stored exponent.

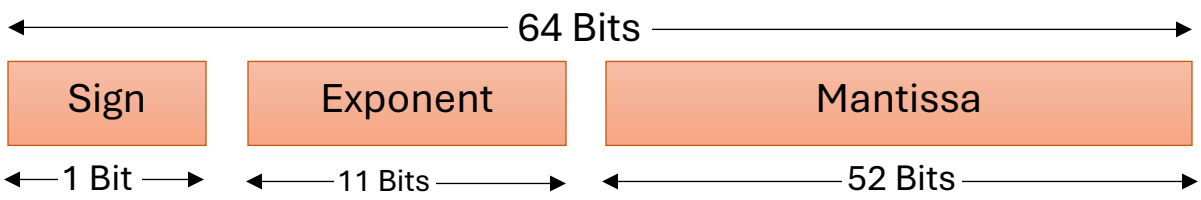
3. **The Normalized Mantissa**: The mantissa is the significant digits of the number in scientific notation. In IEEE 754, the mantissa is normalized so that there is exactly one non-zero digit (1) to the left of the decimal point.

IEEE 754 defines two primary formats based on these components: single precision and double precision.

1. Single Precision



2. Double Precision



Ex1: 75.125 is represented in single precision floating point as below,

- First convert it to binary

  $75.125 = 1001011.001 = 1.001011001 \times 2^6$
- sign = 0
- biased exponent 127 + 6 = 133
- Normalized mantissa = 001011001 , we will add 0's to complete the 23 bits

The IEEE 754 Single precision is:

Sign	Exponent	Mantissa
0	10000101	00101100100000000000000

Advantages and disadvantages of Fixed-point and Floating-point representations

	Advantage	Disadvantage
Fixed Point Representation	The performance is good, not requiring extra hardware or software logic	Limited range of values can represent
Floating point representation	A wider range of numbers can be represented, with different levels of precision.	Requires more storage space, slower processing times, and reduced precision.

Coding systems used in Computers

➔ Binary Coded Decimal (BCD)

BCDis an early coding system used in computing. In this system, each decimal digit is represented by 4 bits. It is designed specifically for representing decimal numbers, allowing up to 16 symbols (since 4 bits can represent 16 different values).

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

Eg: 47

4				7			
0	1	0	0	0	1	1	1

➔ ASCII – American Standard Code for Information Interchange

ASCII, developed by the American National Standards Institute (ANSI), is a 7-bit code used to represent text in computers and communication devices. It's the foundation for most modern character-encoding systems and was the primary encoding system used in IBM PCs.

Extended ASCII expands this by using 8 bits, allowing for 128 additional characters, including special symbols, foreign language letters, and drawing characters.

Eg:

Character	ASCII code	Binary code
null	o	0000000
a	97	1100001
b	98	1100010
0	48	0110000

Character	ASCII code	Binary code
Delete	127	1111111
A	65	1000001
B	66	1000010
1	49	0110001

➔ EBCDIC (Extended Binary Coded Decimal Interchange Code)

This 8-bit character encoding is primarily used on IBM mainframe and midrange systems to represent letters, numbers, and symbols in binary form.

Differences between ASCII and EBCDIC

ASCII	EBCDIC
Used for electronic communication	Used on IBM mainframe & IBM midrange computers
Uses 7 bits	Uses 8 bits
Only 128 characters can be represented	256 characters can be represented
Compatible with Unicode	Not compatible with Unicode
More efficient	Less efficient

➔ UNICODE

Unicode is widely adopted by major companies like Apple, HP, IBM, Microsoft, Oracle, SAP, and many others. It is popular because it can store a vast number of characters, including those from almost every language in the world.

There are different types of Unicode:
- **UTF-8**: Uses 1 byte (8 bits).
- **UTF-16**: Uses 2 bytes (16 bits).
- **UTF-32**: Uses 4 bytes (32 bits).

Unicode typically uses 16 bits, allowing it to represent over 65,000 unique characters, with the ability to support even more.

Differences between ASCII and Unicode

ASCII	Unicode
Needs less space	Needs more space than ASCII
Uses 7 bits	Uses 8/16 or 32 bits
ASCII is a subset of Unicode	Unicode is the superset of ASCII
ASCII characters can be converted to Unicode	Most of the Unicode characters cannot be converted to ASCII
Limited to English	Most of the Written languages can be represented

	Advantage	Disadvantage
BCD	• Easy to encode and decode decimals into BCD and vice versa. • Simple to implement a hardware algorithm for the BCD converter. • It is very useful in digital systems whenever decimal information is given either as inputs or displayed as outputs. • Digital voltmeters, frequency converters and digital clocks all use BCD as they display output information in decimal.	• Not space efficient. • Difficult to represent the BCD form in high speed digital computers in arithmetic operations, especially when the size and capacity of their internal registers are restricted or limited. •Require a complex design of Arithmetic and logic Unit (ALU) than the straight Binary number system. •The speed of the arithmetic operations slow due to the complete hardware circuitry involved.
ASCII	• Uses a linear ordering of letters. • Different versions are mostly compatible. • compatible with modern encodings	• Not Standardized. • Not represent world languages.
EBCDIC	• uses 8 bits while ASCII uses 7 before it was extended. • Contained more characters than ASCII.	• Does not use a linear ordering of letters. • Different versions are mostly not compatible. • Not compatible with modern encodings
UNICODE	• Standardized. • Represents most written languages in the world. • ASCII has its equivalent within Unicode.	• Need twice as memory to store ASCII characters.

Bitwise operations

1. **NOT operation**

   For unsigned integers, the bitwise complement of a number is the "mirror reflection" of the number across the half-way point of the unsigned integers range. One use is to invert a grayscale image where each pixel is stored as an unsigned integer. It uses the below bit operations.

A	NOT A
0	1
1	0

   E.g. :- NOT 0111_2 (7_{10}) = 1000_2 (8_{10})

2. **Bitwise AND operation**

   This is often called bit masking. (By analogy, the use of masking tape covers, or masks, portions that should not be altered or portions that is not of interest. In this case, the 0 values mask the bits that are not of interest.) It uses the below bit operation.

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

   E.g. :- 0101_2 (5_{10}) AND 0011_2 (3_{10})

   $0\ 1\ 0\ 1_2$

   $0\ 0\ 1\ 1_2$

   $0\ 0\ 0\ 1_2$ (1_{10})

   Therefore 0101_2 AND 0011_2 is 0001_2

3. **Bitwise OR operation**

A bitwise OR takes two bit patterns of equal length and performs the logical inclusive OR operation on each pair of corresponding bits. The result in each position is 0 if both bits are 0, while otherwise the result is 1. It uses the below bit operation.

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

E.g. :- 0101_2 (5_{10}) OR 0011_2 (3_{10})

$0\ 1\ 0\ 1_2$

$0\ 0\ 1\ 1_2$

$0\ 1\ 1\ 1_2$ (7_{10})

Therefore 0101_2 OR 0011_2 is 0111_2

4. **Bitwise XOR operation**

The bitwise XOR may be used to invert selected bits in a register (also called toggle or flip). Any bit may be toggled by XOR it with 1 It uses the below bit operations.

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

E.g. :- 0010_2 (2_{10}) XOR 1010_2 (10_{10})

$1\ 0\ 1\ 0_2$

$0\ 0\ 1\ 0_2$

$1\ 0\ 0\ 0_2$ (8_{10})

Therefore 0010_2 XOR 1010_2 is 1000_2

Ex 25 – Compute the Bitwise operations mentioned below.

1) 32 AND 18

2) 21 AND 9

3) 28 AND 13

4) 7 OR 28

5) 16 OR 15

6) 32 OR 57

7) NOT 8

8) NOT 15

9) NOT 18

10) 40 XOR 24

11) 21 XOR 33

12) 53 XOR 35

A/L MCQ QUESTIONS

(2013 A/L) (20) The binary number equivalent to the 25_{10} is

(1) 0100101 (2) 0100111 (3) 0011001 (4) 0010110 (5) 0010111

(2013 A/L) (21) $124_8 + 165_8$ =

(1) 201_8 (2) 289_{10} (3) 289_8 (4) 311_8 (5) 389_8

(2014 A/L) (6) The decimal number equivalent to the 100111_2 is

(1) 40 (2) 39 (3) 38 (4) 37 (5) 36

(2014 A/L) (10) $4A6_{16}+99_{10}$ =

(1) 615_{16} (2) 615_{10} (3) 509_{10} (4) 509_{16} (5) 659_{16}

(2014 A/L) (11) Representations of 5_{10} and -9_{10} in 8-bit Two's complement forms are

(1) 00000101 and 11110111 respectively. (2) 11111011 and 11110111 respectively.

(3) 00000101 and 10001001 respectively. (4) 00000101 and 11110110 respectively.

(5) 11111011 and 11110110 respectively.

(2015 A/L) (3) The decimal number equivalent to 110110_2 is

(1) 39. (2) 48. (3) 54. (4) 55. (5)108.

(2015 A/L) (7) $48B_{16} + 00101011_2$ =

(1) $4B6_{16}$ (2) 310_{16} (3) 503_{16} (4) 513_{16} (5) 559_{16}

(2015 A/L) (42) What is the two's complement representation of 6_{10}?

(1) 11111010 (2) 00000110 (3) 11111001 (4) 01011111 (5) 00000101

(2016 A/L) (3) Which of the following binary numbers is equivalent to 109_{10}?

(1) 1100100_2 (2) 1101101_2 (3) 1001101_2 (4) 1101001_2 (5) 1101100_2

(2016 A/L) (7) $101_{16}+ 110_8$ =

(1) 429_{10} (2) 1011_{10} (3) 329_{10} (4) 529_{10} (5) 137_{10}

(2017 A/L) (3) What is the two's complement representation of 45, if an integer is represented by 8 bits?

(1) 11010011 (2) 10110011 (3) 11001101 (4) 00101111 (5) 00101101

(2017 A/L) (9) Which of the following numbers is equivalent to the binary number 101111,7

(1) 57_8 (2) 57_{16} (3) 57_{10} (4) 59_{16} (5) $5F_{16}$

(2017 A/L) (10) $5D_{16} + 10111_2 =$

(1) 73_8 (2) 75_{16} (3) 116_8 (4) 163_8 (5) 164_8

(2017 A/L) (11) What is the binary representation of 9.25?

(1) 1110011101 (2) 00001001.01 (3) 0000100101 (4) 1000100101 (5) 10001001.01

(2018 A/L) (1) Consider the following three numbers in decimal, octal and hexadecimal notations, respectively.

A-231_{10}

B-347_8

C-$E7_{16}$

Which of the above is/are equivalent to 11100111_2 in binary notation?

(1) A only (2) B only (3) A and Conly (4) B and C only (5) All A, B and C

(2018 A/L) (2) What is the decimal equivalent to the binary 110101.11_2?

(1) 53.00_{10} (2) 53.50_{10} (3) 53.75_{10} (4) 54.25_{10} (5) 54.75_{10}

(2019 A/L) (2) Which of the following represents the bitwise AND operation of the two binary numbers 01010101 and 10101010?

(1) 00000000 (2) 00001111 (3) 11001100 (4) 11110000 (5) 11111111

(2019 A/L) (20) Which of the following represents the result of the binary arithmetic operation of 11001100 - 01010101?

(1) 00110011 (2) 01100110 (3) 01110111 (4) 10011001 (5) 10101010

(2019 A/L) (21) Which of the following statements is/are correct about two's complement?

A - Subtraction is carried out as addition

B - Calculations are more efficient.

C - It is possible to represent negative numbers within the two's complement.

(1) A only (2) B only (3) A and B only (4) B and C only (5) All A, B and C

(2019 A/L) (29) What is the binary equivalent to decimal 54.25 ?

(1) 00011111.11 (2) 00101010.01 (3) 00110110.01 (4) 00111011.1 (5) 00111110.1

(2020 A/L) (5) Consider the two binary numbers P = 10110001 and Q = 01001110. If X P OR Q and Y = P AND Q. what will be the values of X and Y respectively?

(1) 01001110, 10110001 (2) 10110001, 00000000 (3) 10110001, 11111111

(4) 11111111, 00000000 (5) 11111111, 10110001

(2020 A/L) (6) What is the 2's complement of decimal -12?

(1) 00001100 (2) 00110011 (3) 11110011 (4) 11110100 (5) 11111011

(2020 A/L) (7) Which of the following is true about 2's complement?

(1) An extra bit is used to represent the sign.

(2) Makes it possible to build low-cost, high-speed hardware to perform arithmetic operations.

(3) Addition and subtraction are used as two different operations.

(4) Usually represented in hexadecimal number system.

(5) Used in first generation computers to perform logic operations.

(2020 A/L) (7) Consider the character representations in Table 1 and Table 2 given below:

Table 1:

ಕ	ಕಾ	ಕಿ	ಕೀ
0D85	0D86	0D87	0D88

Table 2:

அ	ஆ	இ	ஈ
0B85	0B86	0B87	0B88

In which of the character encoding systems given below, the above characters in Table 1 and Table are represented?

(1) Both Tables 1 and 2: in ASCII (2) Both Tables 1 and 2 in UNICODE

(3) Table 1: in ASCII, Table 2: in UNICODE (4) Table 1: in EBCDIC, Table 2: in ASCII

(5) Table 1: in UNICODE, Table 2: in ASCII

(2021 A/L) (5) What is the correct result of bit-wise XOR operation between the two binary numbers 01011100_2 and 11111001_2 ?

(1) 00000010 (2) 01011000 (3) 01011010 (4) 10100101 (5) 11111101

(2021 A/L) (6) What is the correct 2's complement binary representation of decimal -32 using 8-bits?

(1) 00100000 (2) 10100000 (3) 11011111 (4) 11100000 (5)11100001

(2021 A/L) (7) What is the correct decimal equivalent of hexadecimal 88.8?

(1) 88.5_{10} (2) 88.8_{10} (3) 129.5_{10} (4) 136.5_{10} (5) 136.8_{10}

(2021 A/L) (8) A particular command can be used to output the values of every byte in a file in decimal format. Assume a file contains the following text:

Love trees!

Referring the two **Notes (i)** and **(ii)** given below, select the correct output that will result when the said command is run on that file.

(1)	76	111	118	101	32	116	114	101	101	115	10	
(2)	76	111	118	101	116	114	101	101	115	33	10	
(3)	76	111	118	101	32	116	114	101	101	115	33	10
(4)	108	111	118	101	116	114	101	101	115	33	10	
(5)	108	111	118	101	32	116	114	101	101	115	33	10

Notes:

(i) Some selected rows from the ASCII table are given below:

Decimal	Character
10	(LINE FEED)
32	(SPACE)
33	!
76	L
101	e

Decimal	Character
108	1
111	o
114	r
115	s
116	t
118	v

(ii) The file ends with a LINE FEED character.

(2022 A/L) (4) Which of the following gives the correct results of bit-wise AND and bit-wise OR operations between the two binary numbers 01010100_2 and 11101001_2 respectively?

(1) 01000000_2, 11111101_2 (2) 00000010_2, 10111001_2 (3) 10111101_2, 11001010_2

(4) 11000000_2, 00101100_2 (5) 11111101_2, 01010011_2

(2022 A/L) (5) What is the correct binary equivalent of decimal 12.75_{10}?

(1) 1011.01_2 (2) 1011.11_2 (3) 1100.00_2 (4) 1100.11_2 5) 1100.01_2

(2022 A/L) (6) What is the correct 2's complement binary representation of decimal -41_{10} using 8-bits?

(1) 00101001 (2) 01010110 (3) 10101001 (4) 11010110 (5) 11010111

(2022 A/L) (7) The address of an instruction was shown as 1019 in hexadecimal. What is that address in decimal?

(1) 25 (2) 1249 (3) 4345 (4) 10159 (5) 16249

(2022 A/L) (8) A particular command can be used to output a text file in its binary format.

Assume a file contains the following text:

0 Waste!

Referring the Important notes (1) and (ii) given below, select the correct output that will result when the said command is run on that file.

(1) 00110000 00100000 01010111 01100001 01110011 01110100 01100101 00001010

(2) 00110000 01010111 01100001 01110011 01110100 01100101 00100001 00001010

(3) 00110000 00100000 01010111 01100001 01110011 01110100 01100101 00100001 00001010

(4) 00110000 00100000 01110111 01100001 01110011 01110100 01100101 00100001 00001010

(5) 00110000 00100000 01010111 01100001 01110011 01110100 01100101 00100000 00001010

Important notes:

(i) The file ends with a LINE FEED character,

(ii) Some selected rows from the 7-bit ASCII table are given below:

Character	Binary
(LINE FIELD)	0001010
(SPACE)	0100000
!	0100001
0	0110000
W	1010111

Character	Binary
a	1100001
e	1100101
s	1110011
t	1110100
w	1110111

(2023 A/L) (7) What is the correct binary equivalent of decimal 13.125_{10}?

(1) 1100.001 (2) 1100.100 (3) 1101.001 (4) 1101.100 (5) 1101.101

(2023 A/L) (8) Which of the following are equivalent to octal 674_8?

A - $110\ 111\ 100_2$
B - 444_{10}
C - $2BC_{16}$

(1) A only (2) A and B only (3) A and C only (4) B and C only (5) All A, B and C

(2023 A/L) (9) The address of an instruction was shown as 5A1 in hexadecimal. What is that address in decimal?

(1) 41 (2) 1441 (3) 1457 (4) 2641 (5) 23056

(2023 A/L) (10) A document contains 2048 characters including spaces and line-breaks. How many bits are needed to encode this document in ASCII also using the parity bits?

(1) 2048 (2) 2048 x 2 (3) 2048 x 7 (4) 2048 x 8 (5) 2048/8

(2023 A/L) (11) What is the correct 2's complement binary representation of decimal -49_{10} using 8-bits?

(1) 00110001 (2) 01100010 (3) 10011110 (4) 11001111 (5) 11100010

A/L Structured Essay Questions

(2013 P2 A) (3.a)

(i) Convert 13_{10} and -19_{10} into two's complement numbers. Use 8-bits to represent a number. **(3 marks)**
(ii) Compute $13_{10} - 19_{10}$ by using two's complement numbers obtained in section (i) above and give the answer in two's complement form. **(1 mark)**
(iii) Explain how the positive and negative numbers in Two's complement can be converted into decimal numbers. **(1 mark)**

(2014 P2 A) (2.a) Show that the negative of 0001_2 is 1111_2 Note that both numbers are represented in two's complement form. **(3 marks)**

(2015 A/L P2 A) (2.a) Assume that in a particular digital device, integers are represented in 8-bits two's complement form. However, the results of computations are printed in decimal.

(i) Give the representation of 10_{10} in the above device. **(1 mark)**
(ii) Give the representation of -25_{10} in the above device. **(1 mark)**
(iii) Explain how the computation of $10_{10} - 25_{10}$ done by the device by using your representations given in sections (i) and (ii) above. **(1 mark)**
(iv) List the steps necessary to transform the result obtained in section (iii) above into decimal form in order to print the answer. **(1 mark)**

(2018 A/L P2 A) (b)

(i) Write down the two's complement representation of 12_{10} using 8 bits. **(1 mark)**
(ii) Write down the two's complement representation of -68_{10} using 8 bits. **(1 mark)**
(iii) compute $-68_{10} + 12_{10}$ using the above representations (i) and (ii) **(1 mark)**
(iv) State one advantage of using two's complement representation for data in internal operations of a computer. **(1 mark)**