Copy kar rahe ho toh changes karo properly

**Batch: A4 Roll No.: 16010122083**

**Experiment / assignment / tutorial No.\_\_\_\_\_\_\_**

|  |
| --- |
| **TITLE: Implementation of IEEE-754 floating point representation** |

**AIM:** To demonstrate the single and double precision formats to represent floating point numbers.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment: CO 1**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. Carl Hamacher, ZvonkoVranesic and SafwatZaky, “Computer Organization”, Fifth Edition, TataMcGraw-Hill.
2. William Stallings, “Computer Organization and Architecture: Designing for Performance”, Eighth Edition, Pearson.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Pre Lab/ Prior Concepts:**

The IEEE Standard for Floating-Point Arithmetic (IEEE 754) is a [technical standard](https://en.wikipedia.org/wiki/Technical_standard) for [floating-point](https://en.wikipedia.org/wiki/Floating_point) computation established in 1985 by the [Institute of Electrical and Electronics Engineers](https://en.wikipedia.org/wiki/Institute_of_Electrical_and_Electronics_Engineers) (IEEE). The standard [addressed many problems](https://en.wikipedia.org/wiki/Floating_point" \l "IEEE_754_design_rationale) found in the diverse floating point implementations that made them difficult to use reliably and [portably](https://en.wikipedia.org/wiki/Software_portability). Many hardware [floating point units](https://en.wikipedia.org/wiki/Floating_point_unit) now use the IEEE 754 standard.

The standard defines:

* *arithmetic formats:* sets of [binary](https://en.wikipedia.org/wiki/Binary_code) and [decimal](https://en.wikipedia.org/wiki/Decimal) floating-point data, which consist of finite numbers (including [signed zeros](https://en.wikipedia.org/wiki/Signed_zero) and [subnormal numbers](https://en.wikipedia.org/wiki/Subnormal_number)), [infinities](https://en.wikipedia.org/wiki/Infinity), and special "not a number" values ([NaNs](https://en.wikipedia.org/wiki/NaN))
* *interchange formats:* encodings (bit strings) that may be used to exchange floating-point data in an efficient and compact form
* *rounding rules:* properties to be satisfied when rounding numbers during arithmetic and conversions
* *operations:* arithmetic and other operations (such as [trigonometric functions](https://en.wikipedia.org/wiki/Trigonometric_functions)) on arithmetic formats
* *exception handling:* indications of exceptional conditions (such as [division by zero](https://en.wikipedia.org/wiki/Division_by_zero), overflow, *etc*

**Example (Single Precision- 32 bit representation )**

Represent following real number in single precision format

(23456**.**01171875)10

Out of this the integer part or whole number part can be converted using calculator directly

So (23456)10 = (5BA0)16 = (0101 1011 1010 0000)2

To convert fraction part into binary number we have to do continuous multiplication by 2 and take out integer generated (if any) after every step of multiplication and use remaining fraction in next step.

This process we have to continue till we get fraction as zero OR till the accuracy of given number of binary digits OR when we realize that pattern of ‘0’ & ‘1’ is getting repeated and will result into endless process.

So let us start multiplication process for fraction part from above number-

0

0.01171875 X 2 = 0.0234375

0

0.0234375 X 2 = 0.046875

0

0.046875 X 2 = 0.09375

0

0.09375 X 2 = 0.1875

0

0.1875 X 2 = 0.375

0

0.375 X 2 = 0.75

1

0.75 X 2 = 1.5

1

0.5 X 2 = 1.0

1. Fraction is zero so now we will stop multiplication process

Thus (23456.01171875)10 = (0101 1011 1010 0000 **.** 0000 0011)2

Now let us normalize this number. This means shift binary point to the right of 1st non zero digit. So the number is (01**.** 01 1011 1010 0000 0000 0011 X 214 )2

Exponent E’ = E + 127 = 14 + 127 = (141)10 = (8d)16 = (1000 1101)2

Given number is positive so sign bit is ‘0’

0 1000 1101 01 1011 1010 0000 0000 0011 0

Sign Exponent Mantissa 23 bits

Bit 8 Bits

Let us rewrite above number so that we can express it as a hexadecimal number

0100 0110 1011 0111 0100 0000 0000 0110

(4 6 B 7 4 0 0 6)

**Example (Double Precision- 64 bit representation )**

Represent same above real number in double precision format

(23456**.**01171875)10 = (0101 1011 1010 0000 **.** 0000 0011)2 We have done this conversion above. This number in Normalized form is as follows

(01**.** 01 1011 1010 0000 0000 0011 X 214 )2

For double precision format exponent is in excess 1023 format (11 bits)

Exponent E’ = E + 1023 = 14 + 1023 = (1037)10 = (40d)16 = ( 100 0000 1101)2

0 100 0000 1101 01 1011 1010 0000 0000 0011 0000 0000 0000 0000 0000 0000 0000 00

Sign Exponent Mantissa 52 bits

Bit 11 Bits

Let us rewrite this number and express in hexadecimal format

0100 0000 1101 0110 1110 1000 0000 0000 1100 0000 0000 0000 0000 0000 0000 0000

4 0 d 6 E 8 0 0 C 0 0 0 0 0 0 0

Convert following single precision number to decimal number

(C7A85000)16

First step is write it in binary format

1100 0111 1010 1000 0101 0000 0000 0000

Let us rewrite it

1 1000 1111 010 1000 0101 0000 0000 0000

Sign Exponent Mantissa 23 bits

Bit 8 Bits

Sign bit is ‘1’ so number is negative

Exponent is E’ = E + 127 = (8F)16 = (143)10 So E = 143 – 127 = (16)10

Number is - 1**.**010 1000 0101 0000 0000 0000 X 216

- **000**1 0101 0000 1010 0000**.**0000 000 Preceding 3 zeros added

- 1 5 0 A 0 **.** 0000 000 (Fraction part is zero)

164 X 1 + 163 X 5 + 162 X 0 + 161 X 10 + 160 X 0 = - 86176 **.** 00

**Post Lab Descriptive Questions**

1. **Give the importance of IEEE-754 representation for floating point numbers?**

1. Standardization: IEEE 754 provides a universal format for representing floating-point numbers, ensuring consistency.

2. Precision: It offers precise representations for a wide range of real numbers.

3. Error Handling: The standard defines how rounding and exceptions are managed, enhancing reliability.

4. Interoperability: IEEE 754 facilitates seamless data exchange across different systems.

5. Performance: It optimizes numerical computations, with support from specialized hardware instructions.

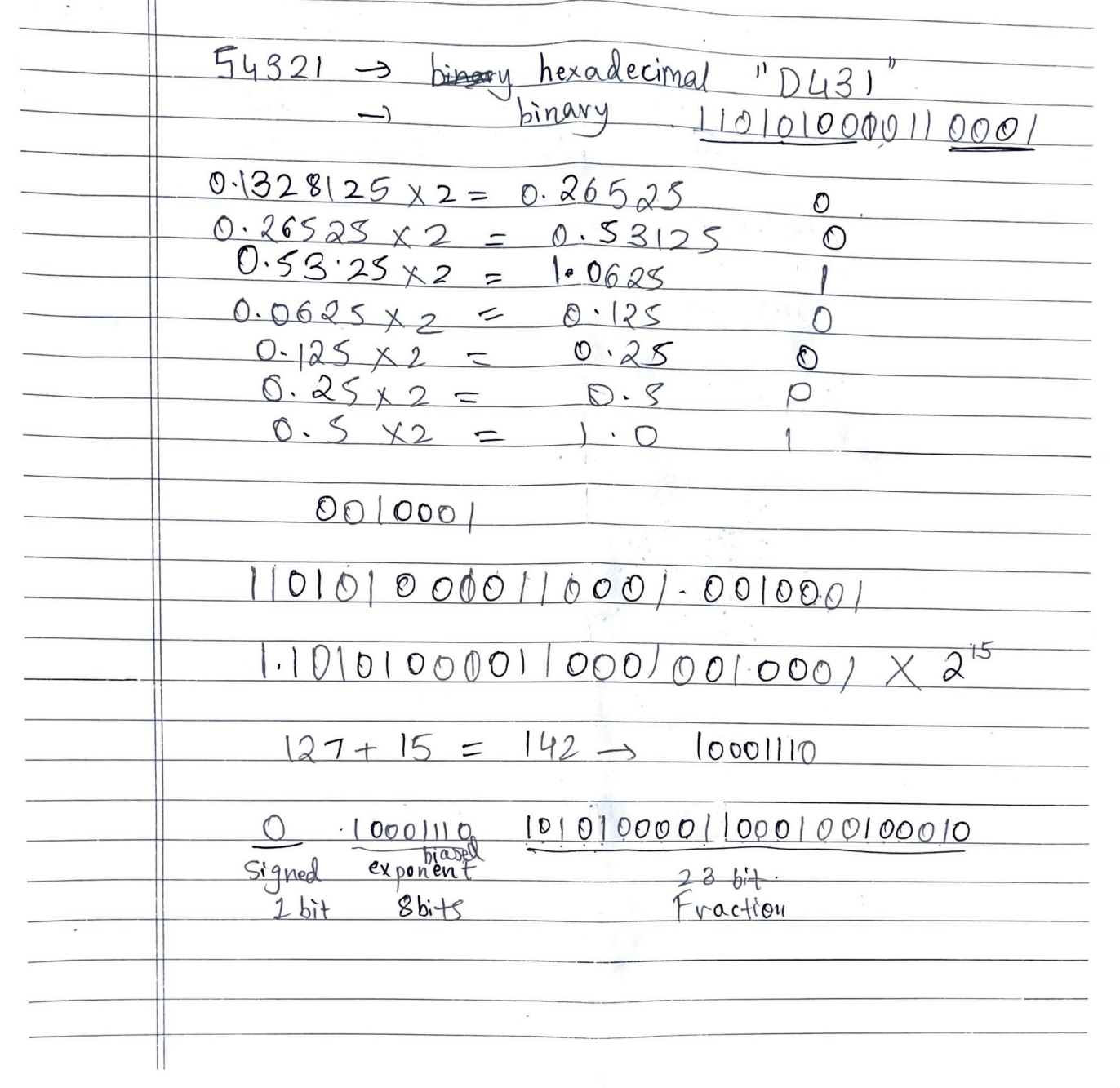
6. Mathematical Consistency: Adherence to mathematical rules ensures predictable results.

7. Widespread Adoption: It's widely used in scientific and engineering fields.

8. Special Values: IEEE 754 defines representations for infinity, NaN, and zero.

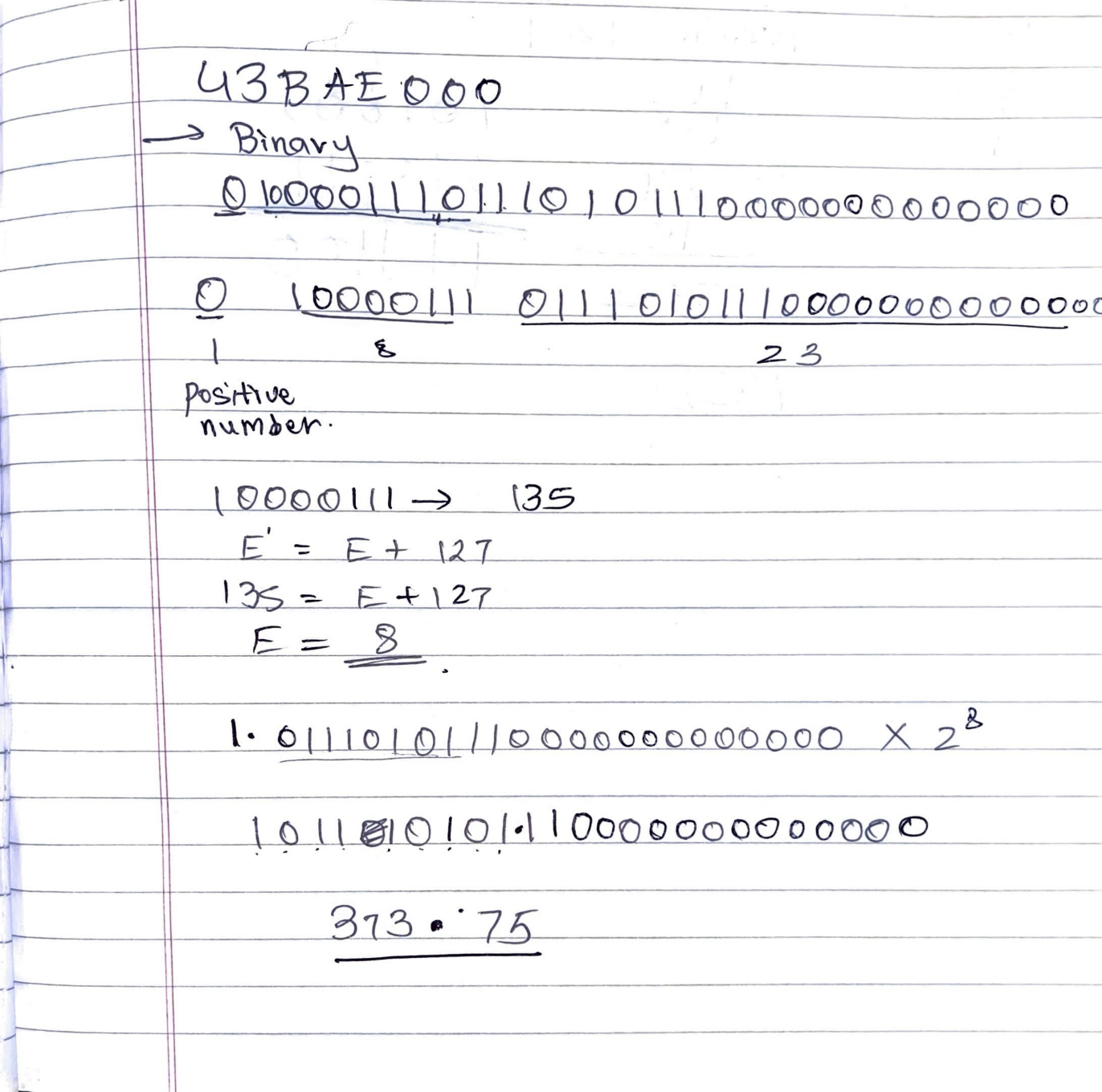
1. Portability: It allows software to be easily moved across platforms.
2. Represent following real number in single precision format

54321**.**1328125 (All the steps similar to above example must be visible in your answer)



1. Convert following single precision number to decimal number

43BAE000 (All the steps similar to above example must be visible in your answer)



**Date: \_\_\_\_\_\_\_\_\_\_\_\_\_**