Batch: D3	**Roll No.:16010123294**
Experiment / assignment / tutorial No. 05	

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: (Mention CO attained here)

Books/ Journals/ Websites referred:

1. Carl Hamacher, Zvonko Vranesic and Safwat Zaky, "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 for floating-point computation established in 1985 by the Institute of Electrical and Electronics Engineers (IEEE). The standard addressed many problems found in the diverse floating point implementations that made them difficult to use reliably and portably. Many hardware floating point units now use the IEEE 754 standard.

The standard defines:

- *arithmetic formats:* sets of binary and decimal floating-point data, which consist of finite numbers (including signed zeros and subnormal numbers), infinities, and special "not a number" values (NaNs)

- *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) on arithmetic formats

- *exception handling:* indications of exceptional conditions (such as division by

zero, overflow, *etc*

Example (Single Precision- 32 bit representation)

```
#include <stdio.h>
#include <stdint.h>
#include <string.h>

void displayBinary(uint32_t number) {
    for (int i = 31; i >= 0; i--) {
        printf("%d", (number >> i) & 1);
        if (i % 8 == 0 && i != 0) {
            printf(" ");
        }
    }
    printf("\n");
}

int main() {
    float inputFloat;
    uint32_t binaryRep;

    printf("Enter a float number: ");
    scanf("%f", &inputFloat);

    memcpy(&binaryRep, &inputFloat, sizeof(binaryRep));

    uint32_t signBit = (binaryRep >> 31) & 1;
    uint32_t exponentBits = (binaryRep >> 23) & 0xFF;
    uint32_t mantissaBits = binaryRep & 0x7FFFFF;

    printf("Input Number: %f\n", inputFloat);
    printf("Binary Representation: ");
    displayBinary(binaryRep);
    printf("Sign: %d\n", signBit);
    printf("Exponent (biased): %d, Unbiased: %d\n", exponentBits, (int)(exponentBits -
127));
    printf("Mantissa: 0x%X\n", mantissaBits);

    return 0;
}
```

```
Binary Representation: 01000001 00100001 10011001 10011010
Sign: 0
Exponent (biased): 130, Unbiased: 3
Mantissa: 0x21999A
```

Example (Double Precision- 64 bit representation)

```
#include <stdio.h>
#include <stdint.h>
#include <string.h>

void displayBinary64(uint64_t number) {
    for (int i = 63; i >= 0; i--) {
        printf("%d", (number >> i) & 1);
        if (i % 8 == 0 && i != 0) {
            printf(" ");
        }
    }
    printf("\n");
}

int main() {
    double inputDouble;
    uint64_t binaryRepresentation;

    printf("Enter a double: ");
    scanf("%lf", &inputDouble);

    memcpy(&binaryRepresentation, &inputDouble, sizeof(binaryRepresentation));

    uint64_t signBit = (binaryRepresentation >> 63) & 1;
    uint64_t exponentBits = (binaryRepresentation >> 52) & 0x7FF;
    uint64_t mantissaBits = binaryRepresentation & 0xFFFFFFFFFFFFF;

    printf("Number: %lf\n", inputDouble);
    printf("Binary Representation: ");
    displayBinary64(binaryRepresentation);
```

```
    printf("Sign: %llu\n", signBit);
    printf("Exponent (biased): %llu, Unbiased: %lld\n", exponentBits, (long long)(exponentBits - 1023));
    printf("Mantissa: 0x%llX\n", mantissaBits);

    return 0;
}
```

```
Enter a double: 1000000000094
Number: 1000000000094.000000
Binary Representation: 01000010 01101101 00011010 10010100 10100010 00001011 11000000 00000000
Sign: 0
Exponent (biased): 1062, Unbiased: 39
Mantissa: 0xD1A94A20BC000
```

Post Lab Descriptive Questions

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

The IEEE-754 representation is crucial for floating-point numbers as it provides a standardized format that ensures consistency and precision across different computing systems. It allows for efficient storage and manipulation of real numbers, handling a wide range of values (both large and small) while maintaining accuracy. The standard splits a floating-point number into three parts: sign, exponent, and mantissa, which helps in representing both very small and very large numbers effectively, crucial for scientific and engineering computations.

Conclusion

Learned to Implement the method OF IEEE 754 floating point representation.

Date: _____