Yeditepe University
Department of Computer Engineering

CSE 323 - Computer Organization

Assignment 2

Due to: 30.11.2023 23:59

In floating point addition/subtraction, due to the shift operations to align the mantissas, some errors may occur. Write a C program to show that, in floating-point arithmetic, the result of (A+B)-B may not always be equal to A. That is, after the following operations, D may not be equal to A.

$$C \leftarrow A+B$$
$$D \leftarrow C-B$$

1. First, select two numbers for A and B such that when you run your code, the result can be calculated without an error (that is D is equal to A).

2. Then, select different numbers for A and B and show that D is not equal to A.

In both cases, print A, B, C and D.

In selecting the numbers A and B, make sure that the numbers can be represented in the binary accurately.
For example:
$(0.6875)_{10}$ can be accurately represented in 32-bit floating format since
$1 * 2^{-1} + 1 * 2^{-3} + 1 * 2^{-4} = 0.5 + 0.125 + 0.0625 = 0.6875 = 0.1011$
Hence, its 32-bit floating point representation would be 0 01111110 01100000000000000000000

However, $(1.36)_{10}$ cannot be accurately represented in 32-bit floating format, since the fractional part 0.36 =0.0101110000101000111… repeats itself. Hence, its 32-bit floating point representation would be 0 01111101 01110 00010100 011110101

Therefore, you must select numbers A and B such that their fractional parts can be accurately represented in 32- bit floating format, as in number $(0.6875)_{10}$.

Some helpful links:
● Online Binary-Decimal Converter http://www.binaryconvert.com/
● Floating Point Number Representation in C
https://www.cprogramming.com/tutorial/floating_point/understanding_floating_point_representation.html

The following code may help you to find the desired numbers:
```
#include <stdio.h>
#include <stdlib.h>
#include <ieee754.h>

int main(){
       float a;
       for(;;){
              printf("\n\nENTER the value of a:\n");
              scanf("%f", &a);
              union ieee754_float *p_a;
              unsigned int a_exp;
              unsigned int a_negative;
              unsigned int a_mantissa;
              p_a = (union ieee754_float*)&a;
              a_exp = p_a->ieee.exponent;
              a_negative = p_a->ieee.negative;
              a_mantissa = p_a->ieee.mantissa;
              printf("exponent of a: %x\n", a_exp);
              printf("negative of a: %x\n", a_negative);
              printf("mantissa of a: %x\n", a_mantissa);
       }
       return 0;
}
```

Submission:
Submit your C program, outputs of the two runs, and your comments on the results. For what kind of numbers inaccurate results are obtained?