ICSI 333 – SYSTEM FUNDAMENTALS
SPRING 2023

Project I

The total grade for the assignment is 100 points.

You must follow the programming and documentation guidelines (see *Programming Assignments Requirements and Recommendations* on Blackboard).

Due date: 11:59 pm Sunday, February 26, 2023.

DESCRIPTION

You must write a C program that performs the conversion of a decimal number into IEEE 754 Single Precision representation. A user inputs the real number, then the program should output three binary strings that show IEEE 754 binary sign bit, 24-bit mantissa (including hidden bit), and binary biased exponent.

The IEEE 754 standard for floating point numbers is explained in the lecture Number Systems available on Blackboard. In addition, you may find useful the instructions given in the article How to Convert a Number from Decimal to IEEE 754 Floating Point Representation or search for more information online.

Special requirements:

- You should not use any standard library function to convert a decimal number into binary format.
- You should not use any bitwise operators.
- Binary numbers in this project should be in the form of an array or string.
- For the sake of simplicity, the absolute value of the user's input should not exceed the maximum value of the signed long integer type.

Work on the project task by task, starting with small numbers. Once you have correctly working code, elaborate on critical points.

SUBMISSION, GRADING, AND ACADEMIC INTEGRITY

The project must be submitted on Blackboard. You have three attempts; please read *Programming Assignments Requirements and Recommendations* on Blackboard for suggested use of the attempts and **submission** package.

Please read *Programming Assignments Requirements and Recommendations* on Blackboard for the **grading** rubric.

Please read *Programming Assignments Requirements and Recommendations* on Blackboard for a strong warning on **cheating**.

RECOMMENDED STEPS OF DEVELOPMENT

TASK #1. INPUT AND CONVERSION TO BINARY

In this part, you implement algorithms for converting real numbers into binary representation (see lecture Number Systems).

Remember that the whole and fractional parts need different algorithms.

The two's complement is not applicable in IEEE 754. The sign bit for mantissa is the most left bit of the representation and can be handled independently.

TASK #2. MANTISSA

In IEEE 754 mantissa is a 24-bit binary number between $(1)_2$ and $(10)_2$ but the first 1 is not shown, so the total number of bits used for mantissa in IEEE 754 is 23.

To avoid ambiguity your program must show 24 bits.

If the total length of the whole and fractional part is more than 24 bits, the mantissa should be rounded to a 24-bit binary number. Note, the rounding may impact the exponent.

TASK #3. EXPONENT

The exponent is a power of two the mantissa should be multiplied by to calculate the value of the IEEE 754 number. It could be positive or negative, but an 8-bit code is a biased exponent which is 127 greater than the actual exponent. For example, to find the value of the IEEE 754 number the mantissa should be multiplied by 2^{-10}, then the biased exponent is -10 + 127 = 117 = $(01110101)_2$.

The exponent of $(11111111)_2$ is reserved to generate an error message. The exponent of $(00000000)_2$ is reserved to represent decimal 0.0 along with the mantissa of all bits equal to 0.

TASK #4. TESTING

Once your program produces the correct results for simple examples, like 1.0, 10.0, 0.1, -5.0, etc., you should give it a real challenge with big numbers and all critical points. Check how the program works for the biggest number possible, the smallest number possible, a number in exponential format, etc.

Feel free to use online resources to check your results, such as this one https://babbage.cs.qc.cuny.edu/ieee-754.old/decimal.html.

PROGRAMMING SUGGESTIONS

- Work on your project step-by-step. If you cannot fulfil all requirements, but successfully solves some tasks your will be graded anyway. If your code does not compile by gcc, you will have a zero or very low grade.
- Use a `char` array to store binary digits. Print arrays when you need output.
- Use function for every logically separated piece of code. For example, converting whole numbers to binary, calculating the mantissa, etc. Pass the array to the functions by reference.
- Remember that your code cannot process any possible number, apply a correctness check for inputs and function results.
- Your program reads inputs from `stdin` (keyboard) and writes outputs to `stdout` (terminal window).
- After each call to the function `printf`, include the following C statement: `fflush(stdout);`.

  Example: `printf("Number = %d\n", number); fflush(stdout);`
- Remember that no bitwise operators or standard library functions converting to binary are allowed in this project.
- **Do not copy any piece of code from online resources or another person. All works will be checked for plagiarism and academic integrity will be strongly enforced.**

EXAMPLE OF PROGRAM EXECUTION

```
% ./p1
Input a real number: -1
The IEEE 754 Single Precision is
Sign: 1
Mantissa (24 bit): 100000000000000000000000
Exponent: 01111111

% ./p1
Input a real number: 35e10
The IEEE 754 Single Precision is
Sign: 0
Mantissa (24 bit): 101000101111101101000000
Exponent: 10100101

% ./p1
Input a real number: 45000000.08
The IEEE 754 Single Precision is
Sign: 0
Mantissa (24 bit): 101010111010100101010000
```

```
Exponent: 10011000

% ./p1
Input a real number: 2e40
The IEEE 754 Single Precision is
Sign: 0
Mantissa (24 bit): 00000000000000000000000
Exponent: 11111111
Overflow!!!
```