

Debugging and Verifying Programs
CS 340d, Fall 2019
Unique Number: 50960

Lab Assignment 3: Floating-Point Rounding
Assigned: Monday, April 22, 2019
Due: Wednesday, May 8, 2019, by 5:00 pm

Introduction

The purpose of this assignment is to become more familiar with floating-point representation. This lab asks you to write C code that implements the rounding (round-to-even) for single-precision, floating-point addition. In this lab, you will need to think very carefully about the answer that must be produced exactly.

Logistics

You are to work alone on this project, but you may discuss programming tricks and bit-manipulation code with your peers. The entire “hand-in” will be electronic. Any clarifications and revisions to the assignment will be posted on our course Web page.

Hand Out Instructions

You will find the file `fplab-handout.tar` referenced on the homework page of the class website. You will need to download this file so you can use its contents.

Start by copying `fplab-handout.tar` to a (protected) directory in which you plan to do your work. Then give the command: `tar xvf fplab-handout.tar`. This will cause a number of files to be unpacked in the directory. **The only file you will be modifying and turning in is `fp-round.c`.**

Looking at the file `fp-round.c` you’ll notice a place to include your name and UTCS UserID into which you should insert the requested identifying information about yourself. Do this right away so you don’t forget.

The `fp-add.c` file contains a skeleton of code to check your solution. Your assignment is to complete the `fp_add` function by adding code to the file `fp-round.c` using only *straight-line* code (i.e., no loops, but single-level IF statements are OK) and a limited number of C arithmetic and logical operators. Specifically, you are *only* allowed to use the following eight operators:

`! ~ & ^ | + << >>`

For this laboratory, you may also use one-level-deep IF statements; there is an example usage in the `fp-add.c` file. You are not allowed to use any constants longer than 8 bits. See the comments in `fp-add.c` for detailed rules and a discussion of the desired coding style.

Rounding Challenge

You have been given a template for a single-precision, floating-point addition instruction. Before there were floating-point co-processors, microprocessors implemented floating-point computations with a collection of integer instructions. We have given you a template for your solution, and a test harness so you may check your solution. All that you need to do is to implement the rounding code. You may assume that both input numbers will be positive and that they will both have an exponent of 0–254; thus, no infinite or NaN inputs will be provided. One or both of the inputs may be denormalized. You need to complete the `fp_add` function so that it correctly implements round to even.

In the `main` procedure, you may wish to reduce the initial value of the `number_of_tests` variable. We will certainly check your code on a very large number of tests, but when getting started such a large value may print a huge amount of output. During our evaluation of your solution, we may also choose to give the random function a different seed.

Do not write lots of code! We were able to achieve the rounding code in 15 statements (about 20 lines of code). The main thing you need to do is to think carefully about how rounding works. Remember, the only time round-to-even is subtle is when the remainder is exactly 1/2 of the least significant answer bit. You will only submit the `fp-round.c` file – put your rounding code there.

Evaluation

Your code will be compiled with GCC and run and tested on one of the public Linux machines. Your score will be computed out of a maximum of 100 points based on the following distribution:

- 40** Correctness of code running on one of the public Linux machines.
- 30** Performance of code, based on number of operators used.
- 30** Description of your solution; this description should be source-code comments.

Regarding performance, our main concern at this point in the course is that you can get the right answer. However, we want to instill in you a sense of keeping things as short and simple as you can.

Your solution should be as clean and straightforward as possible. Your comments should be precise and informative, but they need not be extensive.

Advice

You are welcome to do your code development using any system or compiler you choose. Just make sure that the version you turn in compiles and runs correctly on our UT CS public Linux machines. If it doesn't compile, we can't grade it.

Hand In Instructions

Instructions for submitting your assignment will appear on Piazza.