ECE 472 Homework 2

Drake Vidkjer

October 30, 2017

1 Part 1: Implement frexp function

Reflection For part 1, I was moving pretty smoothly through the implementation until I got to calculating the fraction. The only problem I had before this was trying to figure out how I wanted to represent the double in order to perform bit sifts and masking on it. I tried using a union but decided it was a little clunky for that purpose, so I ended up using a long by initializing it with a point that pointed to the same memory as the double. This was simple, especially since there was only two variables. Once I got that out of the way the next problem was the mantissa/fraction. I remembered from class that the way frexp returns the fraction is only half of the IEEE 754 standard. I was also having trouble figuring out how to convert the mantissa stored in binary into something useful. I then started to think about it a little more and realized that I could shift the binary point by multiplying by 2 , because it is a base to system, same as you can shift a decimal point my multiplying by 10. This really helped and I was finally able to figure out how to interpret the value of the mantissa.

2 Part 2: Floating point operations

Reflection I had a very difficult time trying to work on an implementation of floating point arithmetic in C. I am an EE major, so software is not my strongest skill. I was able to somewhat get an addition operation going, though it does not calculate the correct value. I did set it up against the fpu to test if it was slower, and it was noticeably slower than the floating point unit. I would imagine that this difference in time would only increase with

more complex operations. The way I attempted to implement addition was to find the difference between the exponents of the two values, then shift the smaller value so the exponents lined up, then add the mantissas of the two exponents. If I had more knowledge, I would have also added an overflow so that if the resulting mantissa was more then 53 bits, then the exponent would be increased. Anyways, once the mantissa was added, then the floating point number could be reassembled by combining the mantissa and exponent shifted to the correct positions.

3 Part 3: Feature extraction

Reflection For this part I ended up using a union like the assignment suggested. I had already tried using it a little in part 1, so I already knew a little about how to set it up. A union basically sets up a number of variables so that they all share the same memory space. A union is meant to help conserve memory by using a single memory space for multiple variables, though in this instance I used it for an unintended purpose. There's a few lines near the beginning that give the user the chance to hard code some values in.

References

[1] https://ryanstutorials.net/binary-tutorial/binary-floating-point.php

[2] http://www.cplusplus.com/reference/cmath/frexp/

[3] `https://en.wikipedia.org/wiki/Double-precision_floating-point_format`

[4] https://stackoverflow.com/questions/34558621/get-the-sign-mantissa-and-exponent-of-a-floating-point-number

[5] https://stackoverflow.com/questions/15685181/how-to-get-the-sign-mantissa-and-exponent-of-a-floating-point-number