Computer Science 210 –Computer Organization

Assignment 2

Due 11:59 PM on Monday, Jan 31 2022

I recommend you start early and finish this assignment well in time as this is a rather short assignment and should not take more than a couple of hours. You may discuss things with your assignment buddy, or make use of office hours. If you use any late days for this assignment, make sure to log it on Canvas in the Google Form provided. Once completed, place all required files in a folder, named **assignment_2** and zip it before submitting it on Canvas.

For this assignment, you've already been provided with two tester programs exercises. First, copy your **binary.py** from assignment 1 to current assignment's folder. You should then clean up your binary/integer conversion functions from last week and test them by running **testbinary1.py**.

You should then add the functions in the following exercises, suitably documented, to your file named **binary.py**, and test them with the tester program **testbinary2.py**.

In these exercises, you will extend your library of number system functions to include functions to convert from floating-point numbers in base 10 to IEEE single precision format in binary. You should develop them bottom-up, in the following order:

1. Define a Python function **`unsignedFractionToBinary`** that expects two arguments, an **unsigned** floating-point number (a Python `float` that is less than 1) and a maximum number of bits. This function returns the corresponding string of binary digits (just the digits, no decimal point). Note that the length of the string returned must be less than or equal to the maximum number of bits. Test the function with the example numbers you can find in the book and with 25 bits maximum.

2. Define a Python function **`unsignedFloatToBinary`** that expects two arguments, an **unsigned** floating-point number (a Python **`float`** that can be greater than or equal to 1) and a maximum number of bits for the fractional part. This function returns the corresponding string of binary digits. The returned bit string should include the binary number's whole part, a decimal point, and the fractional part. Test as in Exercise 1.

3. Define a Python function **`normalize`** that expects one argument, a bit string of the form returned by **`unsignedFloatToBinary`**. The **`normalize`** function returns the equivalent bit string in normalized form. Examples of returned values are the string "1.101E-3" from the bit string "0.001101" and the string "1.01100E4" from the bit string "10110.0". Note the embedded exponent, whose value is represented in base 10 digits.

4. Define a Python function **`decimalToSinglePrecision`** that expects one argument, a **signed** floating-point number (a Python **`float`**). This function returns a string of binary digits representing that number in IEEE single precision format. You should make good use of the other functions in your library for this one. Above all, don't reinvent the wheel! As before, consult the class slides for test data.