CSI 250 - Homework Set #5

To receive credit on this assignment, you need to show your work. If you believe there is ”no work to show” then explain how you derive your answer instead. Spacing in binary digits is to help with parsing the long strings of numbers.

IEEE 754 is a standard for floating point numbers that is used in most general purpose modern architectures. In class, we saw a designation for 16-bit floating points that used a 5-bit excess 15 exponent, for 32-bit that used an 8-bit excess 127 exponent and a 64-bit version that used an 11-bit excess 1023 exponent. The remaining bits were used for a sign bit (the first bit) and the fractional part with a hidden 1. If the exponent is all 0s or all 1s we are in a special case that helps represent 0, subnormal numbers, infinities or NaN (not a number). The 16 bit format is sometimes referred to as half precision, 32 bits as single precision and 64 bits as double precision. Refer to the notes from class, for a more complete picture.

1. Conversion Practice You can use some web pages like
   `https://www.h-schmidt.net/FloatConverter/IEEE754.html`
   to check your answers, but you need to show your work for credit.

   (a) Convert $(-312.125)_{10}$ into an IEEE 754 32-bit single precision floating point number.

   (b) Convert
   1 1000 1000 0001 0001 1000 0000 0000 000
   from single precision into decimal.

   (c) Convert
   0 1100 0000 0000 0000 0000 0000 0000 000
   from single precision into decimal.

   (d) Do the same for
   0 1100 0000 0000 0000 0000 0000 0000 001

   (e) Given the challenge of looking at 32 bits all at once, often these formats are displayed to humans in a hexadecimal representation. Due to the direct conversion of binary to hex and vice versa, this means an 8 character hex representation can represent all 32 bits. Convert 0x4b000000 from single precision to decimal.

   (f) Convert 0x4b000001 from single precision to decimal.

2. Examining limits

   (a) What is the largest possible exponent in single precision? Demonstrate its value in bits and in decimal. (Furthest from 0 we can get.)

   (b) What is the smallest possible exponent in single precision? Demonstrate its value in bits and in decimal. (Closest to 0 we can get).

   (c) What is the largest possible number in single precision? Demonstrate its value in bits and in decimal.

3. Examining precision

   (a) Examine the difference between the values in 1.c and 1.d above. Write these in decimal scientific notation.

   (b) How many decimal places is the first difference in values. For example, if you wrote 3.76573 x 10^{12} and 3.76688 x 10^{12} for your answers in the previous step, these numbers differ at the third decimal place. This is a hint at the number of digits of precision in these numbers. In this example, there would be roughly 4 digits of precision. We get this by comparing two values that are "close" in the floating point format. There is no number that is between 1.c and 1.d that is representable by floating point formats.

   (c) What is the difference, in decimal between these two numbers. This gives you an idea of the distance between two digits, but only at this exponent. We'll examine this next.

   (d) Repeat this process for 1.e and 1.f.

   (e) Compare your results for the precision and distance between these two answers. The digits of precision should be similar. But the distance between neighboring values should be much different. What is this demonstrating here?

4. Examining a different format
   Imagine a format similar to IEEE 754 but with one critical difference, only 7 bits are used for the exponent. This would be stored in excess 63. Let's examine how these changes compare to the standard.
   (a) What is the largest exponent in this new format?
   (b) Convert 0x56000000 in this new format to decimal.
   (c) The value of 0x56000001 in decimal is 0.5 higher than the answer in the previous question. How many digits of precision does this format seem to have? The answer here is a bit misleading. The precision is different in these two formats, but further investigation would suggest that they do not differ by an entire point of decimal precision. Remember, these types of investigations are hints at precision rather than true calculations of precision.
   (d) Discuss why this format might be inferior to the format chosen by IEEE 754. In what environment might it be superior?

5. Writing some code
   Use a language of your choice to do the following. Treat the bits as a list (or array or vector or array list as you like). So, when the question refers to an 8 bit number, you can treat it as a list containing 8 total 0s and 1s. The first number in the list will be the most significant bit.
   (a) Given two unsigned 8-bit binary numbers, write an algorithm to compare the two numbers and report which is higher. Hint, work from the most significant bit and move towards least significant bits.
   (b) Given two twos complement 8-bit binary numbers, write an algorithm to compare the two numbers and report which is higher.
   (c) Given two single precision numbers, write an algorithm to compare the two numbers and report which is higher. (You can ignore special cases here, but in practice, positive infinite is bigger than all numbers, negative infinite is smaller than all numbers, NaN fails all tests/comparisons, while lastly, positive and negative 0 are equal.)

6. In one ascii file format that I was decoding, I once saw a number in the file that was: 1079605461 This struck me as very odd, as I was looking for a floating point value of 3.398 that could edit and mess with. I'm not sure if the author of this file type was intentionally being obfuscating, but this definitely took me a moment to decode. This is the process you should take in this problem. First, convert 1079605461 into binary. Then, decode that binary number as if it were a single precision float. What number do you get?

7. As we move from half, to single to double precision, why does the exponent only grow by 3, while the fractional part grows much more (from 10 to 23 to 52). How does the largest number change as we move between these three formats? We've calculated, roughly, the digits of precision for single precision. How does the digits of precision change as we move between these formats? (You may look up these answers or compute them. If you look them up, cite your reference).