CS33 Homework Assignment 3

Due 11:59pm, October 1, 2021

1. We argued in class that one of the benefits of the two's-complement approach for representing numbers is that the computer can do arithmetic without concern for whether unsigned or signed numbers are being used, and the result will be correct in either case. Thus, for example, the Intel x86 instruction set does not have separate signed and unsigned versions of its *add* and *subtract* instructions, but *add* and *subtract* both produce correct results regardless of whether one is interpreting their operands (and results) as signed or unsigned.

   The Intel x86 *multiply* instruction, for 64-bit operands, can produce a 128-bit result, thus eliminating any concern about overflow. However, unlike the *add* and *subtract* instructions, there are both a signed multiply and an unsigned multiply. Explain why. (Hint: consider multiplying a 64-bit value that consists of all ones by a 64-bit encoding of positive 2.)

   Note that similar concerns apply to division, for which there might be a 128-bit dividend and a 64-bit divisor producing a 64-bit quotient and a 64-bit remainder.

2. IEEE floating point represents values closest to zero using its denormalized form. All other values, with the exception of a few special cases, are represented using its normalized form.

   a. Suppose there were no denormalized form, and thus an exponent value of zero would be treated as if the exponent were -1023 and there would be an implied leading one in the significand. What would be the smallest representable positive value, and how much larger would the next smallest be?

   b. What is the minimum number of significant bits in the significand for normalized values? What is the minimum number of significant bits in the significand for denormalized values?