Floating Point Arithmetic Notes

Travis Askham (modified by Niall Mangan)

October 7, 2016

1 Floating point

1.1 Floating point numbers

A double-format number is represented by 64 bits, which we label as

$$\pm \qquad a_1 a_2 \dots a_{11} \qquad b_1 b_2 \dots b_{52}$$

One of the bits is used to determine the sign of the number, eleven bits are used to determine the exponent, and the remaining fifty two bits determine what's called the mantissa. Assuming the exponent is right, the number of correct bits in the mantissa for any given calculation gives a sense of the relative accuracy of that calculation (we also often speak of significant digits or digits of accuracy, meaning the number of correct digits when the number is converted to base 10). For normal floating point numbers, all fifty two bits in the mantissa contain information. This is no longer true for "subnormal" numbers, which we will define below.

The value corresponding to a set of bits can be figured out from the following table

If exponent string is:	Then the value is:	In base 10 with if $b_i = 0\ \forall i$, the value is:
$(00000000000)_2 = (0)_{10}$	$\pm(0.b_1 b_2 \dots b_{52})_2 \times 2^{-1022}$	0
$(00000000001)_2 = (1)_{10}$	$\pm(1.b_1 b_2 \dots b_{52})_2 \times 2^{-1022}$	$\pm 2.2251 \times 10^{-308}$
$(00000000010)_2 = (2)_{10}$	$\pm(1.b_1 b_2 \dots b_{52})_2 \times 2^{-1021}$	$\pm 4.4501 \times 10^{-308}$
$\vdots$	$\vdots$	$\vdots$
$(01111111111)_2 = (1023)_{10}$	$\pm(1.b_1 b_2 \dots b_{52})_2 \times 2^{0}$	± 1
$(10000000000)_2 = (1024)_{10}$	$\pm(1.b_1 b_2 \dots b_{52})_2 \times 2^{1}$	± 2
$\vdots$	$\vdots$	$\vdots$
$(11111111101)_2 = (2045)_{10}$	$\pm(1.b_1 b_2 \dots b_{52})_2 \times 2^{1022}$	$\pm 4.4942 \times 10^{307}$
$(11111111110)_2 = (2046)_{10}$	$\pm(1.b_1 b_2 \dots b_{52})_2 \times 2^{1023}$	$\pm 8.9885 \times 10^{-307}$
$(11111111111)_2 = (2047)_{10}$	$\pm\infty$ if $b_i = 0\ \forall i$, NaN otherwise	$\pm\infty$

A single-format number is represented using 32 bits, which we label as

$$\pm \qquad a_1 a_2 \dots a_8 \qquad b_1 b_2 \dots b_{23}$$

The value corresponding to a set of bits can be figured out from the following table

If exponent string is:	Then the value is:
$(00000000)_2 = (0)_{10}$	$\pm(0.b_1 b_2 \dots b_{23})_2 \times 2^{-126}$
$(00000001)_2 = (1)_{10}$	$\pm(1.b_1 b_2 \dots b_{23})_2 \times 2^{-126}$
$(00000010)_2 = (2)_{10}$	$\pm(1.b_1 b_2 \dots b_{23})_2 \times 2^{-125}$
$\vdots$	$\vdots$
$(01111111)_2 = (127)_{10}$	$\pm(1.b_1 b_2 \dots b_{23})_2 \times 2^{0}$
$(10000000)_2 = (128)_{10}$	$\pm(1.b_1 b_2 \dots b_{23})_2 \times 2^{1}$
$\vdots$	$\vdots$
$(11111101)_2 = (253)_{10}$	$\pm(1.b_1 b_2 \dots b_{23})_2 \times 2^{126}$
$(11111110)_2 = (254)_{10}$	$\pm(1.b_1 b_2 \dots b_{23})_2 \times 2^{127}$
$(11111111)_2 = (255)_{10}$	$\pm\infty$ if $b_i = 0\ \forall i$, NaN otherwise

Note that for either format, the numbers corresponding to the exponent zero are interpreted differently. The leading zero allows these "subnormal" numbers to take very small values, but at the cost of significant digits (i.e. we are "wasting" bits of the mantissa to make a smaller exponent).

1.2 Machine epsilon

The value of machine epsilon is determined by the next largest floating point number after the number 1. For normal numbers, this gives a sense of the relative error in rounding a real number to the nearest floating point number. If y is a real number in the range of the normal floating point numbers, then its floating point representation $\hat{y}$ satisfies $\hat{y} = y(1+\delta)$ for some δ with $|\delta| < \epsilon$.

- Double-format: $\epsilon = 2^{-52} \approx 2.2 \times 10^{-16}$
- Single-format: $\epsilon = 2^{-23} \approx 1.2 \times 10^{-7}$

1.3 Floating point arithmetic

With IEEE correctly rounded arithmetic, the results of arithmetical operations have nice error bounds and behavior. Let round be the function that represents a number as its nearest floating point approximation and a circled operation signify its floating point equivalent. Assume x, y are floating point numbers.

- $x \oplus y = \text{round } (x+y) = (x+y)(1+\delta)$
- $x \ominus y = \text{round } (x-y) = (x-y)(1+\delta)$

- $x \otimes y = \text{round } (x \times y) = (x \times y)(1 + \delta)$
- $x \oslash y = \text{round } (x/y) = (x/y)(1 + \delta)$
- $x \otimes 1 = x$
- $x \ominus y = 0 \Rightarrow x = y$

where $|\delta| < \epsilon$ (assuming that the result is a normal floating point number).

However, it's not all good. Indeed, it is possible that

$$(x \oplus y) \ominus z \neq \text{round } (x + y - z)$$

for x, y, and z floating-point numbers. Consider $x = z = 1$ and $y = (1 + \text{rand}())2^{-52}$ in double-format.

1.4 Some other important terms

- Overflow: when, in the course of a computation, numbers become too large to be represented and are replaced by infinity.
- Underflow: when, in the course of a computation, numbers become to small to be represented with normal numbers (so that they are replaced by subnormals or zero) and significant digits may be lost.

1.5 Qualifying exam-type questions

- Find three double precision IEEE floating point numbers a, b, and c for which the relative error of $a + b + c$ is very large. Try to make it as bad as possible and explain your reasoning. Try where all numbers are normal and where you allow subnormal numbers.
- What is the smallest positive integer which is not exactly represented as a single precision IEEE floating point number? What is the largest finite integer which is part of the double precision IEEE floating point system?
- Find the IEEE single and double precision floating point representations (values of a_i and b_i of the numbers $4, 100, 1/100, 2^{100}, 2^{200}$, and 2^{1050} .
- What is the approximate value of machine epsilon in the IEEE double-format floating-point standard?