PH502: Scientific Programming Concepts

Irish Centre for High End Computing (ICHEC)

September 23, 2020

Overview

- In this lecture we will cover floating point numbers.
- Previously we talked about integer number and arithmetic, which are exact.
- Due to the fact that there are an infinite number of real numbers, floating point numbers have limited precision.
- Not only this but floating point arithmetic is also inexact.

$$1.123 \times 1.123 = 1.261129$$

Floating Point Representation

- IEEE 754 binary floating point standard to represent floating point numbers: *e.g.*
$$((-1)^s \times m \times 2^e)_2.$$
- The sign bit, s, is 0 for positive numbers and 1 for negative numbers.
- The exponent, e, is an integer; it implies finite range.
- m=1.f where f is a binary fraction such that $(1)_2 \leq m < (10)_2$ (in decimal: $1 \leq m < 2$) ; It implies finite precision.
  Example: If $x = -13.125$, then
  Base 10: -1.3125×10 where $s = 1, m = (1.3125)_{10}, e = 1$
  Base 2: $-(1101.001)_2$ where $s = 1, m = (1.101001)_2, e = 3$
- Normalised number: We assume the first bit is 1.

Bit Pattern

- Single precision numbers include an 8-bit exponent field and a 23-bit fraction, for a total of 32 bits.
- Double precision numbers have an 11-bit exponent field and a 52-bit fraction, for a total of 64 bits.
  - The IEEE single precision floating-point representation of x has a precision of 24 binary digits:

$$x = (-1)^s \times (1.m_1 m_2 ... m_{23}) \times 2^e$$

  - The IEEE double precision floating-point representation of x has a precision of 53 binary digits:

$$x = (-1)^s \times (1.m_1 m_2 ... m_{52}) \times 2^e$$

- In decimal representation: Maximum number of decimal digits that can be approximated is $\log_{10}(2^{24}) \approx 7.225$ for single, $\log_{10}(2^{53}) \approx 15.955$ for double precision.
- For single precision $-126 \leq e < 127$; For double precision $-1022 \leq e < 1023$.

Reserved

- Zero is not directly representable in the straight format, due to the assumption of a leading 1. It is defined when an exponent field of all zero bits, and a fraction field of all zero bits: $\pm 0.0000\ldots 2^0$. -0 (negative zero) and $+0$ (positive zero) are distinct values.
- Denormalised numbers are when the exponent is zero but the fraction is not. Then the leading digit is assumed to be zero. This represents a number $(-1)^s \times 0.m_1 m_2 ... \times 2^0$.
- $\pm\infty$ when the exponent are all 1s and fraction of all 0s.Ex: $1.0/0.0, -1.0/0.0$
- NaN when exponent all 1s and non zero fraction; used to represent a value that does not represent a real number. Ex: $0/0$ or the square root of a negative number.

Precision and Accuracy

- Arithmetic with integers is exact, unless the answer is outside the range of integers that can be represented; floating point arithmetic is not exact since some real numbers require an infinite number of digits to be represented.
- Some simple decimal numbers cannot be represented exactly in binary. For example,

$$0.10 = (0.0001100110011...)_2$$

- Even some integers cannot be represented in the IEEE format. For example, int y = 33554431 (when assigned to 4-byte real) will be printed as

$$33554432.000000.$$

<u>Rounding:</u> Assume that $x = 1.m_1 m_2 ... m_n m_{n+1}$ but the floating point representation contains n binary digits.

- If m_{n+1} is 0, chop x to n digits.
- If m_{n+1} is 1, chop x to n digits and add 1 to the last digit of result.

Machine Epsilon

Accuracy: Correctness. For $\pi = 3.14159265359...$, 3.133333333 specifies π with 10 decimal digits of precision and two decimal digits of accuracy. How accurate can a number be stored in the floating point representation?

- The machine epsilon is the difference between 1 and the next larger number that can be stored.
- The smallest floating point number with the property that

$$1 + \epsilon > 1.$$

- In single precision: $\epsilon = 2^{-23} \approx 1.19 \times 10^{-7}$.
- In double precision: $\epsilon = 2^{-52} \approx 2.22 \times 10^{-16}$.