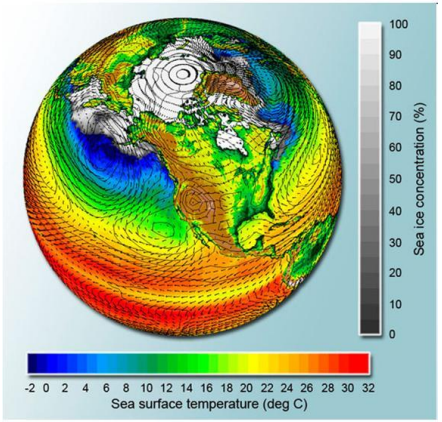
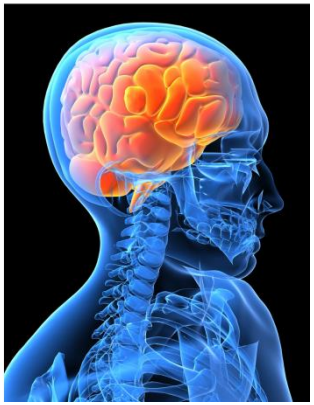
SCHOOL OF COMPUTING
UNIVERSITY OF UTAH

Floating-point Numbers

Zvonimir Rakamarić
University of Utah

FP Computations are Ubiquitous





IEEE 754 Standard

- Well-known floating-point standard
- Published in 1985
- Almost everyone follows it

- So why are we even talking about this?

Challenges

- FP is “weird”
  - Does not faithfully match math (finite precision)
  - Non-associative
  - Heterogeneous hardware support
- FP code is hard to get right
  - Lack of good understanding
  - Lack of good and extensive tool support
- FP software is large and complex
  - High-performance computing (HPC) simulations
  - Stock exchange

FP is Weird

- Finite precision and rounding
  - x + y in reals ≠ x + y in floating-point
- Non-associative
  - (x + y) + z ≠ x + (y + z)
  - Creates issues with
    - Compiler optimizations (e.g., vectorization)
    - Concurrency (e.g., reductions)
- Standard completely specifies only +, -, *, /, comparison, remainder, and sqrt
  - Only recommendation for some functions (trigonometry)

FP is Weird cont.

- Heterogeneous hardware support
  - x + y*z on Xeon ≠ x + y*z on Xeon Phi
    - Fused multiply-add
  - Intel's online article "Differences in Floating-Point Arithmetic Between Intel Xeon Processors and the Intel Xeon Phi Coprocessor"
- Common sense does not (always) work
  - x "is better than" log(e^x)
  - (e^x-1)/x "can be worse than" (e^x-1)/log(e^x)
    - Error cancellation

Hard to Get Right

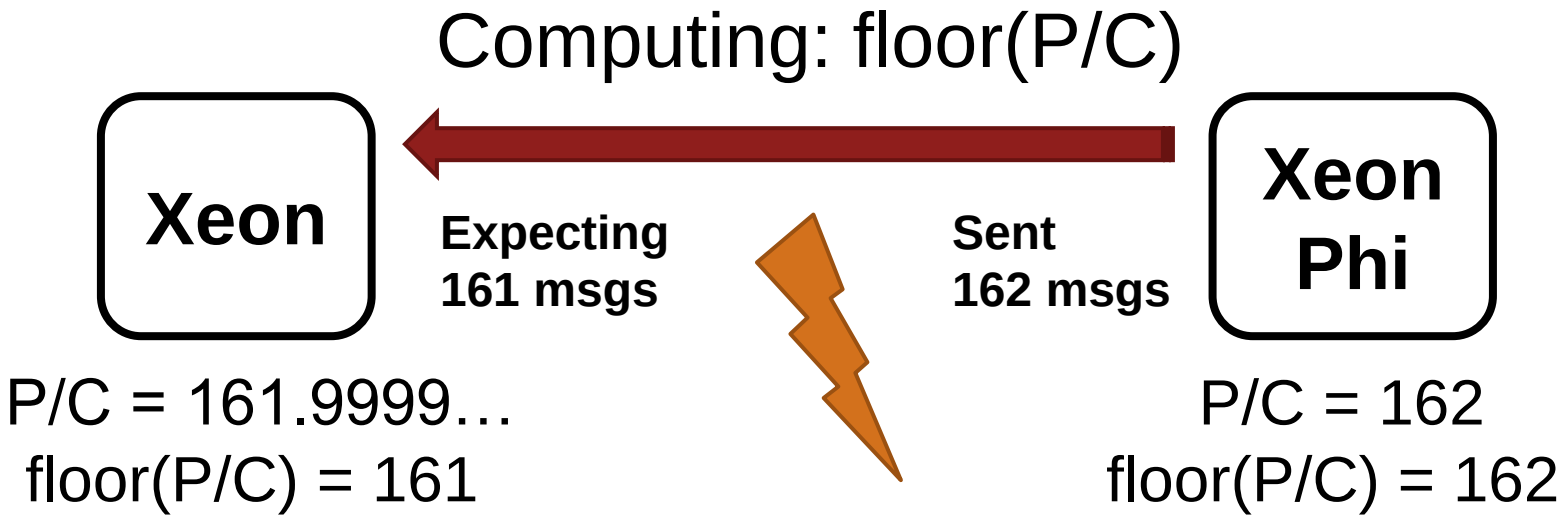
- Think about your triangle classifier
- Poor (no?) tool support
- Pascal Cuoq on John Regehr's blog:
  "The problem with floating-point is that people start with a vague overconfident intuition of what should work, and progressively refine this intuition by removing belief when they are bitten by implementations not doing what they expected."

Hard to Get Right cont.

- Uintah HPC framework developers
  - Advanced, senior, knowledgeable developers
  - Tedious manual debugging to root-cause an FP-related bug
- Personal communication (paraphrasing)
  - “When I turned on vectorization my output suddenly changed.”
  - “My OpenMP program occasionally returns a different output.”
  - “I have no idea what is going on.”

Real-World Examples of Bugs

- Patriot missile failure in 1991 (webpage)
  - Miscalculated distance due to floating-point error
  - Time in tenths of second as measured by the system's internal clock was multiplied by 1/10 to produce the time in seconds
- Inconsistent FP calculations in Uintah



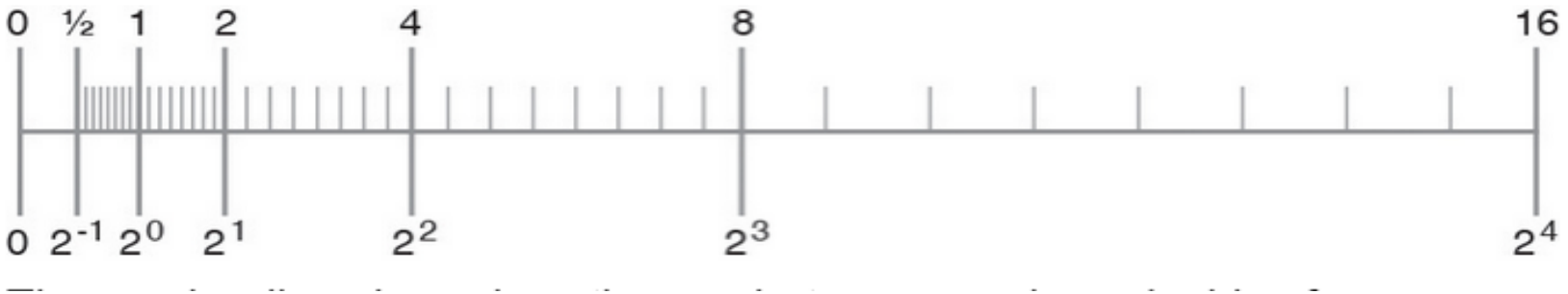
Floating-point Numbers

- Sign, mantissa, exponent
- ((-1)^S) * 1.M * 2^E
- Single precision: 1, 23, 8
- Double precision: 1, 52, 11

Floating-point Number Line

- 3 bits for precision
- Between any two powers of 2, there are $2^3 = 8$ representable numbers

Figure 2-6 Number Line



Error Grows with Magnitude

Table 2-13 Gaps Between Representable Single-Format Floating-Point Numbers

x	nextafter(x, +•)	Gap
0.0	1.4012985e–45	1.4012985e–45
1.1754944e–38	1.1754945e–38	1.4012985e–45
1.0	1.0000001	1.1920929e–07
2.0	2.0000002	2.3841858e–07
16.000000	16.000002	1.9073486e–06
128.00000	128.00002	1.5258789e–05
1.0000000e+20	1.0000001e+20	8.7960930e+12
9.9999997e+37	1.0000001e+38	1.0141205e+31

Rounding Modes

- 4 standard rounding modes
  - Round to nearest (default)
  - Round to 0
  - Round to plus infinity
  - Round to minus infinity
- Can be controlled
- Idea: randomize rounding mode ☺

Floating-Point Operations

- First normalize to the same exponent
  - Larger exponent -> shift mantissa right
- Then perform the operation
- Losing bits when exponents are not the same!
- Example

Kahan Summation

- Smaller error when performing summation
- Cool idea: Keep a separate *running compensation* (a variable to accumulate small errors)
- See wiki for pseudocode