

CSE 160

Lecture 11

Floating point arithmetic

Announcements

- Quiz

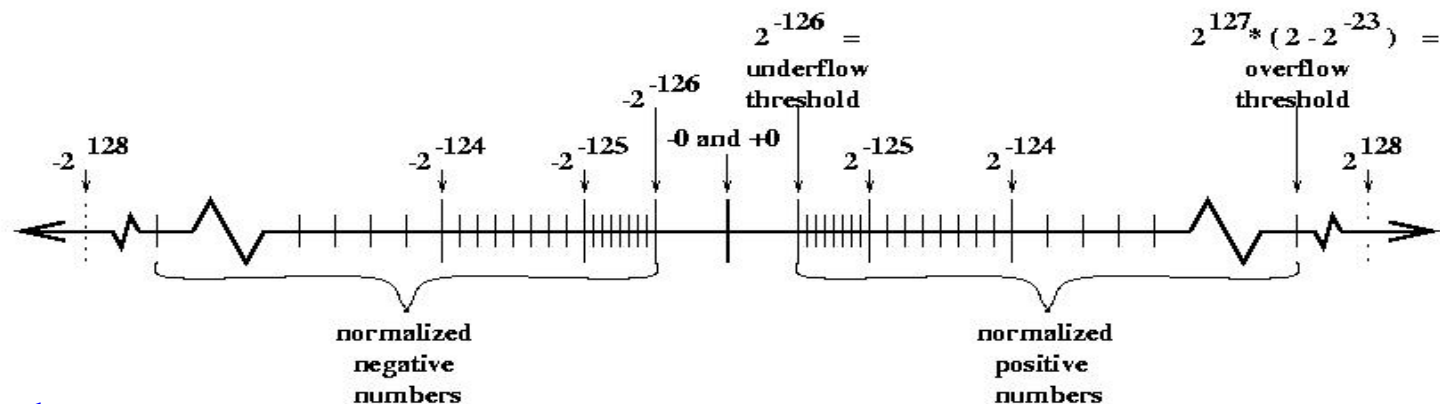
What is floating point?

- A representation
 - $\pm 2.5732... \times 10^{22}$
 - NaN ∞
 - Single, double, extended precision
- A set of operations
 - $+$ $=$ $*$ $/$ $\sqrt{}$ rem
 - Comparison $<$ $\leq$ $=$ $\neq$ $\geq$ $>$
 - Conversions between different formats, binary to decimal
 - Exception handling
- IEEE Floating point standard P754
 - Universally accepted
 - W. Kahan received the Turing Award in 1989 for design of IEEE Floating Point Standard
 - Revision in 2008

IEEE Floating point standard P754

- Normalized representation $\pm 1.d\dots d \times 2^{\text{exp}}$
 - Macheps** = Machine epsilon = $\varepsilon = 2^{-\#\text{significant bits}}$
relative error in each operation
 - OV** = overflow threshold = largest number
 - UN** = underflow threshold = smallest number
- $\pm\text{Zero}$: $\pm\text{significant}$ and exponent = 0

Format	# bits	#significant bits	macheps	#exponent bits	exponent range
Single	32	23+1	2^{-24} ($\sim 10^{-7}$)	8	$2^{-126} - 2^{127}$ ($\sim 10^{+-38}$)
Double	64	52+1	2^{-53} ($\sim 10^{-16}$)	11	$2^{-1022} - 2^{1023}$ ($\sim 10^{+-308}$)
Double	≥ 80	≥ 64	$\leq 2^{-64}$ ($\sim 10^{-19}$)	≥ 15	$2^{-16382} - 2^{16383}$ ($\sim 10^{+-4932}$)

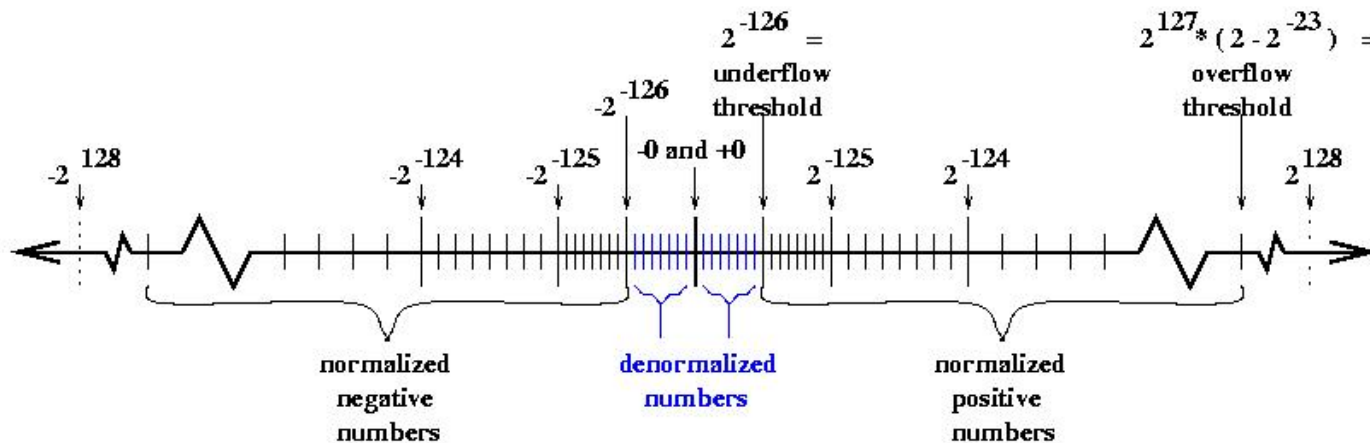


What happens in a floating point operation?

- Round to the nearest representable floating point number that corresponds to the exact value(correct rounding)
- Round to nearest value with the lowest order bit =0 (rounding toward nearest even)
- Others are possible
- We don't need the exact value to work this out!
- Applies to $+$ $=$ $*$ $/$ $\sqrt{}$ rem
- Error formula: $\text{fl}(a \text{ op } b) = (a \text{ op } b) * (1 + \delta)$ where
 - op one of $+$, $-$, $*$, $/$
 - $|\delta| \leq \epsilon$
 - assuming no overflow, underflow, or divide by zero
- Addition example
 - $\text{fl}(\sum x_i) = \sum_{i=1:n} x_i * (1 + e_i)$
 - $|e_i| \sim (n-1)\epsilon$

Denormalized numbers

- Compute: if ($a \neq b$) then $x = a/(a-b)$
- We should never divide by 0, even if $a-b$ is tiny
- *Underflow* exception occurs when exact result $a-b < \text{underflow threshold UN}$
- Return a *denormalized number* for $a-b$
 - Relax restriction that leading digit is 1: $\pm 0.d\dots d \times 2^{\text{min_exp}}$
 - Reserve the smallest exponent value, lose a set of small normalized numbers
 - Fill in the gap between 0 and UN uniform distribution of values



Anomalous behavior

- Floating point arithmetic is not associative

$$(x + y) + z \neq x + (y + z)$$

- Distributive law doesn't always hold
- These expressions have different values when $y \approx z$
 $x * y - x * z \neq x(y - z)$
- Optimizers can't reason about floating point
- If we compute a quantity in extended precision (80 bits) we lose digits when we store to memory $y \neq x$

```
float x, y=..., z=...;  
x = y + z;  
y=x;
```

NaN (Not a Number)

- Invalid exception
 - Exact result is not a well-defined real number

$0/0, \sqrt{-1}$
- NaN op number = NaN
- We can have a quiet NaN or an sNaN
 - Quiet –does not raise an exception, but propagates a distinguished value
 - E.g. missing data: $\max(3, \text{NaN}) = 3$
 - Signaling - generate an exception when accessed
 - Detect uninitialized data

Exception Handling

- An exception occurs when the result of a floating point operation is not representable as a normalized floating point number
 - ▶ $1/0$, $\sqrt{-1}$
- P754 standardizes how we handle exceptions
 - ▶ **Overflow**: - exact result $> OV$, too large to represent
 - ▶ **Underflow**: exact result nonzero and $< UN$, too small to represent
 - ▶ **Divide-by-zero**: nonzero/0
 - ▶ **Invalid**: $0/0$, $\sqrt{-1}$, $\log(0)$, etc.
 - ▶ **Inexact**: there was a rounding error (common)
- Two possible responses
 - ▶ Stop the program, given an error message
 - ▶ Tolerate the exception, possibly repairing the error

An example

- Graph the function

$$f(x) = \sin(x) / x$$

- $f(0) = 1$
- But we get a singularity @ $x=0$: $1/x = \infty$
- This is an “accident” in how we represent the function (W. Kahan)
- We *catch* the exception (divide by 0)
- Substitute the value $f(0) = 1$

Exception handling

- An important part of the standard, 5 exceptions
 - Overflow and Underflow
 - Divide-by-zero
 - Invalid
 - Inexact
- Each of the 5 exceptions manipulates 2 flags
- Sticky flag set by an exception
 - Remains set until explicitly cleared by the user
- Exception flag: should a trap occur?
 - If so, we can enter a trap handler
 - But requires precise interrupts, causes problems on a parallel computer
- We can use exception handling to build faster algorithms
 - Try the faster but “riskier” algorithm
 - Rapidly test for accuracy
(possibly with the aid of exception handling)
 - Substitute slower more stable algorithm as needed