CSE 160
Lecture 19

Floating Point Arithmetic
Supercomputers
Today’s lecture

• Floating Point Arithmetic
• Supercomputing
What is floating point?

• A representation
  ‣ $\pm2.5732\ldots \times 10^{22}$
  ‣ Single, double, extended precision
  ‣ NaN $\infty$

• 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 the design of IEEE Floating Point Standard
  ‣ Revised in 2008
IEEE Floating point standard P754

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

<table>
<thead>
<tr>
<th>Format</th>
<th># bits</th>
<th>#significand bits</th>
<th>macheps</th>
<th>#exponent bits</th>
<th>exponent range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single</td>
<td>32</td>
<td>23+1</td>
<td>$2^{-24} (~10^{-7})$</td>
<td>8</td>
<td>$2^{-126} - 2^{127} (~10^{+38})$</td>
</tr>
<tr>
<td>Double</td>
<td>64</td>
<td>52+1</td>
<td>$2^{-53} (~10^{-16})$</td>
<td>11</td>
<td>$2^{-1022} - 2^{1023} (~10^{+308})$</td>
</tr>
<tr>
<td>Double</td>
<td>≥80</td>
<td>≥64</td>
<td>$\leq 2^{-64 (~10^{-19})}$</td>
<td>≥15</td>
<td>$2^{-16382} - 2^{16383} (~10^{+4932})$</td>
</tr>
</tbody>
</table>

Jim Demmel

©2013  Scott B. Baden / CSE 160 / Fall 2013
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 $+ = * / \sqrt{\text{rem}}$

Error formula: $\text{fl}(a \, \text{op} \, b) = (a \, \text{op} \, b) \times (1 + \delta)$ where
  - $\text{op}$ one of $+, -, *, /$
  - $|\delta| \leq \varepsilon$
  - assuming no overflow, underflow, or divide by zero

Addition example
  - $\text{fl}(\sum x_i) = \sum_{i=1:n} x_i \times (1+e_i)$
  - $|e_i| \sim < (n-1)\varepsilon$
Denormalized numbers

- Compute: if \( a \neq b \) then \( x = \frac{a}{a-b} \)
- We should never divide by 0, even if \( a-b \) is tiny
- Underflow exception occurs when exact result \( a-b < \) underflow threshold UN
- Return a denormalized number for \( a-b \)
  - Relax restriction that leading digit is 1: \( \pm 0.d\ldots d \times 2^{\text{min-exp}} \)
  - 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 \(y \neq x\)

```c
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,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, √-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, √-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
  
  \[ f(x) = \frac{\sin(x)}{x} \]

- \( f(0) = 1 \)
- But we get a singularity \( @ x=0: \frac{1}{x} = \infty \)
- 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
Why is it important to handle exceptions properly?

- Crash of Air France flight #447 in the mid-atlantic
- Flight #447 encountered a violent thunderstorm at 35000 feet and super-cooled moisture clogged the probes measuring airspeed
- The autopilot couldn’t handle the situation and relinquished control to the pilots
- It displayed the message “Invalid Data” without explaining why
- Without knowing what was going wrong, the pilots were unable to correct the situation in time
- The aircraft stalled, crashing into the ocean 3 minutes later
- At 20,000 feet, the ice melted on the probes, but the pilots didn't’t know this so couldn’t know which instruments to trust or distrust.
Today’s lecture

• Floating Point Arithmetic

• Supercomputing
What is the world’s fastest supercomputer?

- Top500 #1, Tianhe-2 @ NUDT (China)
Tianhe -2

• Developed at National University of Defense Technology in Changsha, China
• #1 on Top500.org
• 3.12 Million cores
• 54.9 Tflop/sec peak
• 17.8 MW power (+6MW for cooling)
• 1 PB memory (2^{50} Bytes)
• Kylin Linux
Why numerically intensive applications?

- Highly repetitive computations are prime candidates for parallel implementation
- Improve quality of life, economically and technologically important
  - Data Mining
  - Image processing
  - Simulations – financial modeling, weather, biomedical
- We can classify applications according to Patterns of communication and computation that persist over time and across implementations
  Phillip Colella’s 7 Dwarfs

Courtesy of Randy Bank
Classifying the application domains

- Patterns of *communication* and computation that persist over time and across implementations
  - Structured grids
    - Panfilov method
  - Dense linear algebra
    - Matrix multiply, Vector-Mtx Mpy
    - Gaussian elimination
  - N-body methods
  - Sparse linear algebra
    - In a sparse matrix, we take advantage of knowledge about the locations of non-zeros, improving some aspect of performance
  - Unstructured Grids
  - Spectral methods (FFT)
  - Monte Carlo

Courtesy of Randy Bank
Application-specific knowledge is important

• There currently exists no tool that can convert a serial program into an efficient parallel program

  ... for all applications ... all of the time ... on all hardware

• The more we know about the application...

  ... specific problem ... math/physics ... initial data ...

  ... context for analyzing the output...

  ... the more we can improve productivity

• Issues
  ‣ Data motion and locality
  ‣ Load balancing
  ‣ Serial sections
Up and beyond to Exascale

• In 1961, President Kennedy mandated a landing on the Moon by the end of the decade
• July 20, 1969 at tranquility base “The Eagle has landed”
• The US Govt set an ambitious schedule to reach $10^{18}$ flops by 2018, x1000 performance increase
• DOE is taking the lead in the US, EU also engaged
• Massive technical challenges
The Challenges to landing “Eagle”

• High levels of parallelism within and across nodes
  $10^6$ devices. $10^9+$ threads
• Power consumption: $\gtrsim 20$ MW @ Exascale, but
  nearly at 10MW today, with “just” 0.02 ExaFlops
  Data storage & access consumes most of the energy
• Processor technology hasn’t settled
  ‣ Mixture of accelerators and conventional cores
  ‣ NUMA processors, not fully cache coherent on-chip
• Data motion costs continue to grow
  ‣ Complicated memory hierarchies
  ‣ Hide latency, conserve locality
• Software, software, software
  ‣ Code become more complicated, in order to effectively
    utilize the hardware’s capabilities
35 years of processor trends

Original data collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten
Dotted line extrapolations by C. Moore
Technological trends

- Growth: cores/socket rather than sockets
- Memory/core is shrinking
- Complicated software-managed parallel memory hierarchy
- Cost of communication increasing relative to computation
Data motion costs are rising

- Increase amount of computation performed per unit of communication
  - Conserve locality, tolerate or avoid communication
- Many threads
Hybrid processing

- Two types of processors: general purpose + accelerator
  - AMD fusion: 4 x86 cores + hundreds of Radeon GPU cores
- Accelerator can perform certain tasks more quickly than the conventional cores
- Accelerator amplifies relative cost of communication
Heterogeneous processing with Graphical Processing Units

- Specialized *many-core* processor
- Explicit data motion
  - between *host* and *device*
  - inside the device
Graphical Processing Units

- Processes long vectors
- Thousands of highly specialized cores
- NVIDIA, AMD
NVIDIA GeForce GTX 280

- Hierarchically organized clusters of streaming multiprocessors
  - 240 cores @ 1.296 GHz
  - Peak performance 933.12 Gflops/s
- 1 GB “device” memory (frame buffer)
- 512 bit memory interface @ 132 GB/s

GTX 280: 1.4B transistors
Intel Penryn: 410M (110mm²) (dual core)
Nehalem: 731M (263mm²)
Blue Gene

- IBM-US Dept of Energy collaboration
- Low power, high performance interconnect
- 3rd Generation: Blue Gene/Q
- Sequoia at LLNL, #3 on top 500
  - 16.3 Tflops, 7.89 MW
  - 1.6M cores/93,304 compute nodes, 16 GB mem/node
  - 20 Petaflops = 20,000 TeraFlops = 20M GFlops
  - 96 racks, 3,000 square feet
  - 64-bit PowerPC (A2 core)
  - Weighs about the same as 30 elephants
Hierarchical Packaging

1. Chip
   Single Chip
   16 cores

2. Module
   Single Chip

3. Compute Card
   One single chip module,
   16 GB DDR3 Memory

4. Node Card
   32 Compute Cards,
   Optical Modules, Link Chips, Torus

5a. Midplane
   16 Node Cards

5b. I/O Drawer
   8 I/O Cards
   8 PCIe Gen2 slots

6. Rack
   2 Midplanes
   1, 2 or 4 I/O Drawers

7. System
   96 racks, 20PF/s

SC ‘10, via IBM

©2013 Scott B. Baden / CSE 160 / Fall 2013
Blue Gene/Q Interconnect

- 5D toroidal mesh (end around)
  - Can scale to > 2M cores
  - 2 GB/sec bidirectional bandwidth (raw)
    on all 10 links, 1.8 GB available to the user
  - 5D nearest neighbor exchange
    \(~1.8\text{GB/s/link, 98\% efficiency}\)
  - Hardware latency ranges from 80 ns to 3\(\mu\)s

- Collective network
  - Global Barrier, Allreduce, Prefix Sum
  - Floating point reductions at \(~95\%\) of peak
Die photograph

- 16 (user) + 1 (OS) cores
- 18th redundant core
- Each core 4-way multithreaded
- 1.6 GHz
- Shared L2$: 32MB
- Network routing integrated on-chip
- Compare with layout of the Cray 1 (1976)
The future?

- Lower power is inevitable
- NUMA likely

http://www.hector.ac.uk/cse/documentation/Phase2b/#arch
Example of NUMA - Cray XE6 node

- 24 cores sharing 32GB main memory
- Packaged as 2 AMD Opteron 6172 processors “Magny-Cours”
- Each processor is a directly connected Multi-Chip Module: two hex-core dies living on the same socket
- Each die has 6MB of shared L3, 512KB L2/core, 64K L1/core
  - 1MB of L3 is used for cache coherence traffic
  - Direct access to 8GB main memory via 2 memory channels
  - 4 Hyper Transport (HT) links for communicating with other dies
- Asymmetric connections between dies and processors

www.nersc.gov/users/computational-systems/hopper/configuration/compute-nodes/
XE-6 Processor memory interconnect (node)

- Asymmetric connections between dies and processors
  - Direct access to 8GB main memory via 2 memory channels
  - 4 Hyper Transport (HT) links for communicating with other dies

http://www.ector.ac.uk/cse/documentation/Phase2b/#arch