New-School Machine Structures
(It’s a bit more complicated!)

- Parallel Requests
  Assigned to computer
  e.g., Search “Katz”
- Parallel Threads
  Assigned to core
  e.g., Lookup, Ads
- Parallel Instructions
  >1 instruction @ one time
  e.g., 5 pipelined instructions
- Parallel Data
  >1 data item @ one time
  e.g., Add of 4 pairs of words
- Hardware description
  All gates @ one time

Review
- Flynn Taxonomy of Parallel Architectures
  - SIMD: Single Instruction Multiple Data
  - MIMD: Multiple Instruction Multiple Data
  - SISD: Single Instruction Single Data
  - MISD: Multiple Instruction Single Data
- Intel SSE SIMD Instructions
  - One instruction fetch that operates on multiple operands simultaneously
  - 128/64 bit XMM registers
- SSE Instructions in C
  - Embed the SSE machine instructions directly into C programs through use of intrinsics
  - Achieve efficiency beyond that of optimizing compiler

Agenda
- Amdahl’s Law
- Administrivia
- SIMD and Loop Unrolling
- Technology Break
- Memory Performance for Caches
- Review of 1st Half of 61C

Big Idea: Amdahl’s (Heartbreaking) Law
- Speedup due to enhancement E is
  \[ \text{Speedup w/ E} = \frac{\text{Exec time w/o E}}{\text{Exec time w/ E}} \]
- Suppose that enhancement E accelerates a fraction \( F \) of the task by a factor \( S \), and the remainder of the task is unaffected
  \[ \text{Execution Time w/ E} = \text{Execution Time w/o E} \times (1-F) + F/S \]
  \[ \text{Speedup w/ E} = \frac{1}{1/(1-F) + F/S} \]
Big Idea: Amdahl’s Law

Speedup = \frac{1}{(1-F) + \frac{F}{S}}

Example: the execution time of half of the program can be accelerated by a factor of 2. What is the program speed-up overall?

Example #1: Amdahl’s Law

Speedup w/E = 1 / (1-F + F/S)

• Consider an enhancement which runs 20 times faster but which is only usable 25% of the time.
  Speedup w/E = 1/(.75 + .25/20) = 1.31

• What if its usable only 15% of the time?
  Speedup w/E = 1/(.85 + .15/20) = 1.17

• Amdahl’s Law tells us that to achieve linear speedup with 100 processors, none of the original computation can be scalar!
  To get a speedup of 90 from 100 processors, the percentage of the original program that could be scalar would have to be 0.1% or less.
  Speedup w/E = 1/(.001 + .999/100) = 90.99

Parallel Speed-up Example

\[ \sum_{i=1}^{10} x_i + \sum_{j=1}^{10} y_j \]

• 10 “scalar” operations (non-parallelizable)
• 100 parallelizable operations
• 110 operations
  – 100/110 = .909 Parallelizable, 10/110 = 0.91 Scalar
Example #2: Amdahl’s Law

\[
\text{Speedup} = \frac{1}{\frac{1}{f} + \frac{1}{1-f}}
\]

- Consider summing 10 scalar variables and two 10 by 10 matrices (matrix sum) on 10 processors
  \[\text{Speedup} = \]

- What if there are 100 processors?
  \[\text{Speedup} = \]

- What if the matrices are 100 by 100 (or 10,010 adds in total) on 10 processors?
  \[\text{Speedup} = \]

- What if there are 100 processors?
  \[\text{Speedup} = \]

Strong and Weak Scaling

- To get good speedup on a multiprocessor while keeping the problem size fixed is harder than getting good speedup by increasing the size of the problem.
  - Strong scaling: when speedup can be achieved on a parallel processor without increasing the size of the problem (e.g., 10x10 Matrix on 10 processors to 100)
  - Weak scaling: when speedup is achieved on a parallel processor by increasing the size of the problem proportionally to the increase in the number of processors (e.g., 10x10 Matrix on 10 processors \(\Rightarrow\) 33x33 Matrix on 100)

- Load balancing is another important factor: every processor doing same amount of work
  - Just 1 unit with twice the load of others cuts speedup almost in half

61C in the News

“Remapping Computer Circuitry to Avert Impending Bottlenecks.”

Markoff, NY Times, Feb 28, 2011

Hewlett-Packard researchers have proposed a fundamental restructuring of the modern computer for the coming era of nanoelectronics — a marriage of memory and computing power that could drastically limit the energy used by computers.

Today the microprocessor is in the center of the computing universe, and information is moved, at heavy energy cost, first to be used in computation and then stored. The new approach would be to marry processing to memory to cut down transportation of data and reduce energy use.

The semiconductor industry has long warned about a set of impending bottlenecks described as “the wall,” a point in time where more than five decades of progress in continuously shrinking the size of transistors used in computation will end.

- Systems will be based on memory chips he calls “memistores,” as distinct from the hybrid circuits in which lower-level circuits will be based on a nanoelectronic technology called the memristor, which Hewlett-Packard is developing to store data.

- The memistore chips will have a multistore design, and computing circuits will sit directly on top of the memory to process the data, with minimal energy costs.

Peer Review

- Suppose a program spends 80% of its time in a square root routine. How much must you speedup square root to make the program run 5 times faster?
  \[\text{Speedup} = \frac{1}{\frac{1}{f} + \frac{1}{1-f}}\]

  **A red**: 4
  **B orange**: 5
  **C green**: 10
  **D yellow**: 20
  **E pink**: None of the above

Administrivia

- Lab #7 posted
  - No Homework, no project this week!
- TA Review: Su, Mar 6, 2-5 PM, 2050 VLSB
- Midterm Exam: Tu, Mar 8, 6-9 PM, 145/155 Dwinelle
  - Split: A-Lew in 145, Li-Z in 155
  - Small number of special consideration cases, due to class conflicts, etc. —contact Dave or Randy
- No discussion during exam week; no lecture that day
- Sent (anonymous) 61C midway survey before Midterm: Please fill out! (Only 1/3 so far; have your voice heard!)
- https://www.surveymonkey.com/s/qs3ZLW7

Cut-off for the top 10% is 30 total points. All others will be graded on participation.

Send questions to CS61C@aol.com

E = \frac{1}{(1-F) + E/S}
Getting to Know Profs
• Ride with sons in MS Charity Bike Ride every September since 2002
  • "Waves to Wine"
  • If want to join team, let me know
  • Always a Top 10 fundraising team despite small size
• I was top fundraiser 2006, 2007, 2008, 2009, 2010 due to computing
  – Can offer fundraising advice: order of sending, when to send during week, who to send to...

Agenda
• Amdahl’s Law
• Administrivia
• SIMD and Loop Unrolling
• Technology Break
• Memory Performance for Caches
• Review of 1st Half of 61C

Data Level Parallelism and SIMD
• SIMD wants adjacent values in memory that can be operated in parallel
• Usually specified in programs as loops
  for(i=1000; i>0; i=i-1)
    x[i] = x[i] + s;
• How can reveal more data level parallelism than available in a single iteration of a loop?
  • Unroll loop and adjust iteration rate

Looping in MIPS
Assumptions:
- R1 is initially the address of the element in the array with the highest address
- F2 contains the scalar value s
- 8(R2) is the address of the last element to operate on.
CODE:
Loop: 1. ld F0, 0(R1) ; F0=array element
      2. add.d F4, F0, F2 ; add s to F0
      3. s.d F4, 0(R1) ; store result
      4. addui R1, R1, #-32 ; decrement pointer 8 byte
      5. bne R1, R2, Loop ; repeat loop if R1 != R2

Loop Unrolled
NOTE:
1. Different Registers eliminate stalls
2. Only 1 Loop Overhead every 4 iterations
3. This unrolling works if loop_limit(mod 4) = 0

Loop Unrolled Scheduled
Loop Unrolling in C

- Instead of compiler doing loop unrolling, could do it yourself in C
  for(i=1000; i>0; i=i-1)
  
  x[i] = x[i] + s;
- Could be rewritten
  What is downside of doing it in C?
  for(i=1000; i>0; i=i-4) {
    x[i] = x[i] + s;
    x[i-1] = x[i-1] + s;
    x[i-2] = x[i-2] + s;
    x[i-3] = x[i-3] + s;
  }

Generalizing Loop Unrolling

- A loop of \( n \) iterations
- \( k \) copies of the body of the loop

Then we will run the loop with 1 copy of the body \( n \text{mod} k \) times and with \( k \) copies of the body floor\( (n/k) \) times
- (Will revisit loop unrolling again when get to pipelining later in semester)

Agenda

- Amdahl's Law
- Administrivia
- SIMD and Loop Unrolling
- Memory Performance for Caches
- Technology Break
- Review of 1st Half of 61C

Reading Miss Penalty: Memory Systems that Support Caches

- The off-chip interconnect and memory architecture affects overall system performance in dramatic ways

One word wide organization (one word wide memory)
Assume
- 1 memory bus clock cycle to send address
- 15 memory bus clock cycles to get the 1st word in the block from DRAM (row cycle time), 5 memory bus clock cycles for 2nd, 3rd, 4th words (subsequent column access time)—note effect of latency!
- 1 memory bus clock cycle to return a word of data

Memory-Bus to Cache bandwidth
- Number of bytes accessed from memory and transferred to cache/CPU per memory bus clock cycle

One Word Wide Bus, One Word Blocks

- If block size is one word, then for a memory access due to a cache miss, the pipeline will have to stall for the number of cycles required to return one data word from memory
- memory bus clock cycle to send address
- memory bus clock cycles to read DRAM
- memory bus clock cycle to return data
total clock cycles miss penalty
- Number of bytes transferred per clock cycle (bandwidth) for a single miss is
  bytes per memory bus clock cycle
One Word Wide Bus, One Word Blocks

- If block size is one word, then for a memory access due to a cache miss, the pipeline will have to stall for the number of cycles required to return one data word from memory:
  - 1 memory bus clock cycle to send address
  - 15 memory bus clock cycles to read DRAM
  - 1 memory bus clock cycle to return data
  - 17 total clock cycles miss penalty
- Number of bytes transferred per clock cycle (bandwidth) for a single miss is $4/17 \approx 0.235$ bytes per memory bus clock cycle.

One Word Wide Bus, Four Word Blocks

- What if the block size is four words and each word is in a different DRAM row?
  - 1 cycle to send 1st address
  - 4 x 15 = 60 cycles to read DRAM
  - 1 cycle to return last data word
  - 62 total clock cycles miss penalty
- Number of bytes transferred per clock cycle (bandwidth) for a single miss is $(4 x 4)/62 \approx 0.258$ bytes per clock.

One Word Wide Bus, Four Word Blocks

- What if block size is four words and all words are in the same DRAM row?
  - 1 cycle to send 1st address
  - 15 x 3 x 5 = 225 cycles to read DRAM
  - 1 cycle to return last data word
  - 227 total clock cycles miss penalty
- Number of bytes transferred per clock cycle (bandwidth) for a single miss is $(4 x 4)/227 \approx 0.053$ bytes per clock.

Interleaved Memory, One Word Wide Bus

- For a block size of four words
  - cycle to send 1st address
  - cycles to read DRAM banks
  - cycles to return last data word
  - total clock cycles miss penalty
- Number of bytes transferred per clock cycle (bandwidth) for a single miss is $17 \times 4 = 68$ bytes per clock.
Interleaved Memory, One Word Wide Bus

- For a block size of four words
  - 1 cycle to send 1st address
  - 15 cycles to read DRAM banks
  - 4*4 = 16 cycles to return last data word
  - 20 total clock cycles miss penalty

- Number of bytes transferred per clock cycle (bandwidth) for a single miss is
  - (4 x 4)/20 = 0.8 bytes per clock

DRAM Memory System Observations

- It’s important to match the cache characteristics
  - Caches access one block at a time (usually more than one word)
- 1) With the DRAM characteristics
  - Use DRAMs that support fast multiple word accesses, preferably ones that match the block size of the cache
- 2) With the memory-bus characteristics
  - Make sure the memory-bus can support the DRAM access rates and patterns
  - With the goal of increasing the Memory-Bus to Cache bandwidth

New-School Machine Structures (It’s a bit more complicated!)

- Software
  - Parallel Requests
    - Assigned to computer
    - e.g., Search “Katz”
  - Parallel Threads
    - Assigned to core
    - e.g., Lookup, Ads
  - Parallel Instructions
    - >1 instruction @ one time
    - e.g., 3 pipelined instructions
  - Parallel Data
    - >1 data item @ one time
    - e.g., Add of 4 pairs of words
  - Hardware descriptions
    - All gates functioning in parallel at the same time

Agenda

- Amdahl’s Law
- Adminstrivia
- SIMD and Loop Unrolling
- Memory Performance for Caches
- Technology Break
- Review of 1st Half of 61C

Great Idea #1: Levels of Representation/Interpretation

1. Layers of Representation/Interpretation
2. Moore’s Law
3. Principle of Locality/Memory Hierarchy
4. Parallelism
5. Performance Measurement & Improvement
6. Dependability via Redundancy
Great Idea #4: Parallelism

- Data Level Parallelism in 1st half 61C
  - Lots of data in memory that can be operated on in parallel (e.g., adding together 2 arrays)
  - Lots of data on many disks that can be operated on in parallel (e.g., searching for documents)
- 1st project: DLP across 10s of servers and disks using MapReduce
- Next week's lab, 3rd project: DLP in memory

Summary

- Amdhal’s Cruel Law: Law of Diminishing Returns
- Loop Unrolling to Expose Parallelism
- Optimize Miss Penalty via Memory system
- As the field changes, cs61c has to change too!
- Still about the software-hardware interface
  - Programming for performance via measurement!
  - Understanding the memory hierarchy and its impact on application performance
  - Unlocking the capabilities of the architecture for performance: SIMD