CS 61C
Great Ideas in Computer Architecture
(a.k.a. Machine Structures)
Lecture 1: Course Introduction

Instructors:
Senior Lecturer SOE Dan Garcia (call me “Dan”)
Professor Michael Lustig (call me “Miki”)
(lots of help from TAs, esp Head TA Sagar Karandikar)

http://inst.eecs.berkeley.edu/~cs61c/
Pop Quiz

Q: Anybody know what's up with Professor Lustig getting added to CS 61C?

A: He's an assistant professor, came to berkeley in 2010, so it might be a "get experience" thing
Agenda

• Thinking about Machine Structures
• Great Ideas in Computer Architecture
• What you need to know about this class
Agenda

• Thinking about Machine Structures
• Great Ideas in Computer Architecture
• What you need to know about this class
CS61C is NOT about C Programming

• It is about the hardware-software interface
  – What does the programmer need to know to achieve the highest possible performance

• Languages like C are closer to the underlying hardware, unlike languages like Scheme, Python, Java!
  – We can talk about hardware features in higher-level terms
  – Allows programmer to explicitly harness underlying hardware parallelism for high performance
Old School CS61C
New School CS61C (2/3)
New School CS61C (3/3)

My other computer is a data center
Old School Machine Structures

CS61C

Software

Hardware

Application (ex: browser)

Compiler

Assembler

Operating System

(Mac OSX)

Processor

Memory

I/O system

Datapath & Control

Digital Design

Circuit Design

transistors
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 descriptions**
  All gates working in parallel at same time
Agenda

• Thinking about Machine Structures
• Great Ideas in Computer Architecture
• What you need to know about this class
6 Great Ideas in Computer Architecture

1. Abstraction (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 #1: Abstraction
(Levels of Representation/Interpretation)

<table>
<thead>
<tr>
<th>High Level Language Program (e.g., C)</th>
<th>Compiler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assembly Language Program (e.g., MIPS)</td>
<td>Assembler</td>
</tr>
<tr>
<td>Machine Language Program (MIPS)</td>
<td></td>
</tr>
</tbody>
</table>

Machine Interpretation

Hardware Architecture Description (e.g., block diagrams)

Architecture Implementation

Logic Circuit Description (Circuit Schematic Diagrams)

```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

```
lw $t0, 0($2)
lw $t1, 4($2)
sw $t1, 0($2)
sw $t0, 4($2)
```

Anything can be represented as a number, i.e., data or instructions:

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```
#2: Moore’s Law

Gordon Moore
Intel Cofounder
B.S. Cal 1950!

Predicts:
2X Transistors / chip
every 2 years
Jim Gray’s Storage Latency Analogy: How Far Away is the Data?

- **On Chip Cache** (ns) 1
- **On Board Cache** 2
- **Memory** 100
- **Disk** $10^6$
- **Tape/Optical Robot** $10^9$

- **This Room** 10 min
- **This Campus** 1.5 hr
- **Sacramento** 2 Years
- **Andromeda** 2,000 Years

**Jim Gray**
- Turing Award
- B.S. Cal 1966
- Ph.D. Cal 1969
Great Idea #3: Principle of Locality/Memory Hierarchy
Great Idea #4: Parallelism
Fig 3  Amdahl's Law an Obstacle to Improved Performance  Performance will not rise in the same proportion as the increase in CPU cores. Performance gains are limited by the ratio of software processing that must be executed sequentially. Amdahl's Law is a major obstacle in boosting multicore microprocessor performance. Diagram assumes no overhead in parallel processing. Years shown for design rules based on Intel planned and actual technology. Core count assumed to double for each rule generation.

Caveat! Amdahl’s Law
Great Idea #5: Performance Measurement and Improvement

• Matching application to underlying hardware to exploit:
  – Locality
  – Parallelism
  – Special hardware features, like specialized instructions (e.g., matrix manipulation)

• Latency
  – How long to set the problem up
  – How much faster does it execute once it gets going
  – It is all about time to finish
Coping with Failures

• 4 disks/server, 50,000 servers
• Failure rate of disks: 2% to 10% / year
  – Assume 4% annual failure rate
• On average, how often does a disk fail?
  a) 1 / month
  b) 1 / week
  c) 1 / day
  d) 1 / hour
Coping with Failures

- 4 disks/server, 50,000 servers
- Failure rate of disks: 2% to 10% / year
  - Assume 4% annual failure rate
- On average, how often does a disk fail?
  a) 1 / month
  b) 1 / week
  c) 1 / day
  d) 1 / hour

\[ 50,000 \times 4 = 200,000 \text{ disks} \]
\[ 200,000 \times 4\% = 8000 \text{ disks fail} \]
\[ 365 \text{ days} \times 24 \text{ hours} = 8760 \text{ hours} \]
Great Idea #6: Dependability via Redundancy

- Redundancy so that a failing piece doesn’t make the whole system fail.

Increasing transistor density reduces the cost of redundancy.

2 of 3 agree

FAIL!
Great Idea #6: Dependability via Redundancy

- Applies to everything from datacenters to storage to memory to instructors
  - Redundant datacenters so that can lose 1 datacenter but Internet service stays online
  - Redundant disks so that can lose 1 disk but not lose data (Redundant Arrays of Independent Disks/RAID)
  - Redundant memory bits of so that can lose 1 bit but no data (Error Correcting Code/ECC Memory)
Agenda

• Thinking about Machine Structures
• Great Ideas in Computer Architecture
• What you need to know about this class
Yoda says...

“Always in motion, the future is...”

Our schedule may change slightly depending on some factors. This includes lectures, assignments & labs...
Hot off the presses

• Everyone (on the waitlist), consider telling TeleBears you’re moving to a more open section. We should be able to accommodate everyone, based on past experience.

• **Come to labs and discussion this week**
  – Switching Sections: if there’s room (confirmed by TA in person), go ahead
  – Partners on ALL PROJECTS and LABS
## Weekly Schedule

<table>
<thead>
<tr>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
</tr>
</thead>
<tbody>
<tr>
<td>8AM</td>
<td></td>
<td></td>
<td>LAB 14 - TBD</td>
<td>330 Soda</td>
</tr>
<tr>
<td>9AM</td>
<td>DIS 114 - TBD</td>
<td>76 Evans</td>
<td>LAB 25 - TBD</td>
<td>273 Soda</td>
</tr>
<tr>
<td>10AM</td>
<td>DIS 125 - TBD</td>
<td>67 Evans</td>
<td>LAB 15 - TBD</td>
<td>330 Soda</td>
</tr>
<tr>
<td>11AM</td>
<td>DIS 126 - TBD</td>
<td>4 Evans</td>
<td>LAB 26 - TBD</td>
<td>273 Soda</td>
</tr>
<tr>
<td>12PM</td>
<td>DIS 132 - TBD</td>
<td>105 Latimer</td>
<td>LAB 27 - TBD</td>
<td>277 Soda</td>
</tr>
<tr>
<td>1PM</td>
<td>DIS 131 - TBD</td>
<td>3 Evans</td>
<td>LAB 23 - TBD</td>
<td>330 Soda</td>
</tr>
<tr>
<td>2PM</td>
<td>Lecture</td>
<td>Wheeler Auditorium</td>
<td>LAB 34 - TBD</td>
<td>277 Soda</td>
</tr>
<tr>
<td>3PM</td>
<td>DIS 128 - TBD</td>
<td>4 Evans</td>
<td>Lecture</td>
<td>Wheeler Auditorium</td>
</tr>
<tr>
<td>4PM</td>
<td>DIS 131 - TBD</td>
<td>3 Evans</td>
<td>LAB 28 - TBD</td>
<td>273 Soda</td>
</tr>
<tr>
<td>5PM</td>
<td>DIS 122 - TBD</td>
<td>8 Evans</td>
<td>LAB 18 - TBD</td>
<td>330 Soda</td>
</tr>
<tr>
<td>6PM</td>
<td>DIS 130 - TBD</td>
<td>6 Evans</td>
<td>LAB 29 - TBD</td>
<td>273 Soda</td>
</tr>
<tr>
<td>7PM</td>
<td>DIS 120 - TBD</td>
<td>81 Evans</td>
<td>LAB 19 - TBD</td>
<td>330 Soda</td>
</tr>
<tr>
<td>8PM</td>
<td>LAB 13 - TBD</td>
<td>330 Soda</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9PM</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Course Information

• Course Web: http://inst.eecs.Berkeley.edu/~cs61c/
• Instructors:
  – Dan Garcia & Miki Lustig
• Teaching Assistants: (see webpage)
• Textbooks: Average 15 pages of reading/week (can rent!)
  – Patterson & Hennessey, Computer Organization and Design, 5/e
    (we’ll try to provide 4th Ed pages, not Asian version 4th edition)
  – Barroso & Holzle, The Datacenter as a Computer, 1st Edition
• Piazza:
  – Every announcement, discussion, clarification happens there
Course Grading

- EPA: Effort, Participation and Altruism (5%)
- Homework (10%)
- Labs (5%)
- Projects (20%)
  1. Non-Parallel Application (MIPS & C)
  2. Data Parallelism (Map-Reduce on Amazon EC2)
  3. Parallelize Project1, SIMD, MIMD
  4. Computer Processor Design (Logisim)
- Midterm (25%): 7th & 8th week Friday in class, can be clobbered!
- Final (35%): 2014-12-16 @ 7-10pm
- Performance Competition for honor (and EPA)
Tried-and-True Technique: Peer Instruction

• Increase real-time learning in lecture, test understanding of concepts vs. details
• As complete a “segment” ask multiple choice question
  – 1-2 minutes to decide yourself
  – 2 minutes in pairs/triples to reach consensus.
  – Teach others!
  – 2 minute discussion of answers, questions, clarifications
• You can get transmitters from the ASUC bookstore
  OR you can use i>clicker GO app for less!
  – We’ll start this next week
EECS Grading Policy

- [http://www.eecs.berkeley.edu/Policies/ugrad.grading.shtml](http://www.eecs.berkeley.edu/Policies/ugrad.grading.shtml)
  “A typical GPA for courses in the lower division is 2.7. This GPA would result, for example, from 17% A's, 50% B's, 20% C's, 10% D's, and 3% F's. A class whose GPA falls outside the range 2.5 - 2.9 should be considered atypical.”

- Fall 2010: GPA 2.81
  26% A's, 47% B's, 17% C's, 3% D's, 6% F's

- Job/Intern Interviews: They grill you with technical questions, so it’s what you say, not your GPA (New 61C gives good stuff to say)

<table>
<thead>
<tr>
<th>Year</th>
<th>Fall</th>
<th>Spring</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010</td>
<td>2.81</td>
<td>2.81</td>
</tr>
<tr>
<td>2009</td>
<td>2.71</td>
<td>2.81</td>
</tr>
<tr>
<td>2008</td>
<td>2.95</td>
<td>2.74</td>
</tr>
<tr>
<td>2007</td>
<td>2.67</td>
<td>2.76</td>
</tr>
</tbody>
</table>
Our goal as instructors

• To make your experience in CS61C as enjoyable & informative as possible
  – Humor, enthusiasm & technology-in-the-news in lecture
  – Fun, challenging projects & HW
  – Pro-student policies (exam clobbering)

• To maintain Cal & EECS standards of excellence
  – Projects & exams will be as rigorous as every year.

• To be HKN “7.0” men
  – Dan: I know I speak fast when I get excited about material. I’m told every semester. Help me slow when I go toooo fast.
  – Please give feedback so we can improve!
    Why are we not 7.0 for you? We will listen!!
Extra Credit: EPA!

• **Effort**
  – Attending prof and TA office hours, completing all assignments
  – turning in HW0

• **Participation**
  – Attending lecture and voting using the clickers
  – Asking great Qs in discussion/lecture & making it interactive

• **Altruism**
  – Helping others in lab or on Piazza
  – Writing software, creating art, tutorials that help others learn

• **EPA! extra credit can bump students up to the next grade level**
  – (but EPA! #s are internal)
Late Policy ... Slip Days!

• Assignments due at 11:59:59 PM
• You have 3 slip day tokens (NOT hour or min)
• Every day your project or homework is late (even by a minute) we deduct a token
• After you’ve used up all tokens, it’s 33% off per day.
  – No credit if more than 3 days late
  – Save your tokens for projects, worth more!!
• No need for sob stories, just use a slip day!
Policy on Assignments and Independent Work

- **ALL PROJECTS WILL BE DONE WITH A PARTNER**
- With the exception of laboratories and assignments (projects and HW) that explicitly permit you to work in groups, all homework and projects are to be YOUR work and your work ALONE.
- **PARTNER TEAMS MAY NOT WORK WITH OTHER PARTNER TEAMS**
- **You are encouraged to help teach other to debug.** Beyond that, we don’t want you sharing approaches or ideas or code or whiteboarding with other students, since sometimes the point of the assignment WAS the “algorithm” and if you share that, they won’t learn what we want them to learn. HKN and tutoring sessions that work you through the pseudocode are not allowed. The pseudocode is sometimes the entire point! Feel free to answer questions on Piazza that help them debug (don’t share code, even snippets there). We expect that what you hand in is yours.
- It is NOT acceptable to copy solutions from other students.
- It is NOT acceptable to copy (or start your) solutions from the Web.
- It is NOT acceptable to leave your code anywhere where an unscrupulous student could find and steal it (e.g., public GITHUBs, walking away while leaving yourself logged on, leaving printouts lying around, etc)
- We have tools and methods, developed over many years, for detecting this. You WILL be caught, and the penalties WILL be severe. If you have questions whether a behavior is crossing the line, ask!
- **At the minimum F in the course,** and a letter in your Cal record documenting the incidence of cheating.
- (We’ve caught people in recent semesters!)
- **Both Giver and Receiver are equally culpable and suffer equal penalties**
Architecture of a typical Lecture

Time (minutes)

Clickers

Administrivia

And in conclusion…

Attention

Full

10 30 35 58 60
Summary

• CS61C: Learn 6 great ideas in computer architecture to enable high performance programming via parallelism, not just learn C
  1. Abstraction (Layers of Representation / Interpretation)
  2. Moore’s Law
  3. Principle of Locality/Memory Hierarchy
  4. Parallelism
  5. Performance Measurement and Improvement
  6. Dependability via Redundancy