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/

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 (1/3)

Personal Mobile Devices

New School CS61C (2/3)

Old School Machine Structures

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 Assigned to core
  e.g., Search "Katz"
• Parallel Data Assigned to core
  e.g., Search “Katz”
• Hardware descriptions
  e.g., Physiology and anatomy

Project 1
Project 2
Project 3
Project 4
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)

High Level Language Program (e.g., C)
Assembly Language Program (e.g., MIPS)
Machine Language Program (MIPS)

Compiler
Assembler
Machine Interpretation

Hardware Architecture Description (e.g., block diagrams)
Software Architecture Description (e.g., flow charts)
Logic Circuit Description (Circuit Schematic Diagrams)

Great Idea #2: Moore’s Law

Predicts: 2X Transistors / chip every 2 years

- Gordon Moore
- Intel Co-founder
- B.S. Cal 1950

Great Idea #3: Principle of Locality/ Memory Hierarchy

Jim Gray’s Storage Latency Analogy: How Far Away is the Data?

- Registers
- On Chip Cache
- On Board Cache
- Memory
- Disk
- Tape/ Optical
- Robot
- Tape / Optical

- 1 min
- 1.5 hr
- 2 Years
- 10 min
- 100 Memory
- 10 Tape / Optical
- 109 Tape / Optical

Great Idea #4: Parallelism

- Caveat!
- Amdahl’s Law

Gene Amdahl
Computer Pioneer

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

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

- 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 so that can lose 1 bit but no data (Error Correcting Code/ECC Memory)

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

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...

Course Information

- Instructors: Dan Garcia & Miki Lustg
- Teaching Assistants: (see webpage)
- Textbooks: Average 15 pages of reading/week (can rent!)
  - Patterson &Hennessy, 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
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). You are encouraged to help teach others to debug ALL PROJECTS WILL BE DONE WITH A PARTNER, since sometimes the point of the assignment was the “algorithm” and if homework and projects are to be YOUR work and your work ALONE.

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 transmitted from the ASUC bookstore OR you can use iClicker GO app for less!**
  - We’ll start this next week.

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 and making it interactive
- **Altruism**
  - Helping others in lab or on Piazza
  - Writing software, creating art, tutorials that helps 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**
- **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!**

EECS Grading Policy

- [http://www.eecs.berkeley.edu/Policies/grad_sdgug.pdf](http://www.eecs.berkeley.edu/Policies/grad_sdgug.pdf)
  - 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
  - 28% 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)

Policy on Assignments and Independent Work

- All projects will be done with a partner.
- Full credit will be awarded if work is completed independently.
- A token will be deducted if your work is not independent.
- No need for sob stories, just use a slip day!
- 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!**

Architecture of a typical Lecture

<table>
<thead>
<tr>
<th>Time (minutes)</th>
<th>Full</th>
<th>Clickers</th>
<th>“And in conclusion...”</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>58</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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