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

Course Introduction  

Instructors:  
Mike Franklin  
Dan Garcia  

http://inst.eecs.berkeley.edu/~cs61c/fa11

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 really 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!
  – Allows us to talk about key hardware features in higher level terms
  – Allows programmer to explicitly harness underlying hardware parallelism for high performance
Old School CS61c

Personal Mobile (kinda)New School CS61c (1)
New School CS61c (2)

Old-School Machine Structures

CS61c

Application (ex: browser)
Operating System (Mac OSX)
Instruction Set Architecture

Compiler
Assembler
Processor
Memory
I/O system
Datapath & Control
Digital Design
Circuit Design
transistors

Software
Hardware
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 functioning 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. 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: 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)**
- **Logic Circuit Description (Circuit Schematic Diagrams)**

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101
0101 1000 0000 1001 1100 0110
```

- `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.
Predicts: 2X Transistors / chip every 2 years

#2: Moore’s Law

Curves shows Moore’s Law: transistor count doubling every two years

Gordon Moore
Intel Cofounder
B.S. Cal 1950!

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

Andromeda
$10^9$
Tape/Optical Robot
2,000 Years

Pluto
$10^6$
Disk
2 Years

Sacramento
$100$
Memory
1.5 hr

This Campus
$10$
On Board Cache
10 min

This Room
2
On Chip Cache
1 min

My Head
1
Registers

Jim Gray
Turing Award
B.S. Cal 1966
Ph.D. Cal 1969!
Great Idea #3: Principle of Locality/Memory Hierarchy

Great Idea #4: Parallelism

Jane
Research Composing Typing

Sue
Research Composing Typing

Tom
Research Composing Typing

8/27/11
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

50,000 x 4 = 200,000 disks
200,000 x 4% = 8000 disks fail
365 days x 24 hours = 8760 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

1+1=2
1+1=2
1+1=1

2 of 3 agree

FAIL!

Great Idea #6: 
Dependability via Redundancy

• Applies to everything from datacenters to storage to memory
  – 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

“Always in motion is the future...”

Yoda says...

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

- Due to high student demand, we’ve added a tenth section!!
- It’s the same time as lab 105
- Everyone (not just those on the waitlist), consider moving to this section
Course Information

- Instructors: Dan Garcia, Michael Franklin
- Teaching Assistants: Brian Gawalt (Head TA), Eric Liang, Paul Ruan, Sean Soleyman, Anirudh Todi, and Ian Vonseggern
- Textbooks: Average 15 pages of reading/week (can rent!)
  - Barroso & Holzle, *The Datacenter as a Computer*, 1st Edition
- Piazza:
  - Every announcement, discussion, clarification happens there

Reminders

- Discussions and labs will be held next week
  - Switching Sections: if you find another 61C student willing to swap discussion (from the Piazza thread) AND lab, talk to your TAs
  - Partners (only project 2,3 and performance competition)
Course Organization

• Grading
  – EPA: Effort, Participation and Altruism (5%)
  – Homework (10%)
  – Labs (5%)
  – Projects (20%)
    1. Computer Instruction Set Simulator (C)
    2. Data Parallelism (Map-Reduce on Amazon EC2)
    3. Performance Tuning of a Parallel Application/Matrix Multiply using cache blocking, SIMD, MIMD (OpenMP)
    4. Computer Processor Design (Logisim)
  – Matrix Multiply Competition for honor (and EPA)
  – Midterm (25%): date TBA, can be clobbered!
  – Final (35%): 3-6 PM Thursday December 15th

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 web>clicker app for $10!
  – We’ll start this on Monday
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></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>

Extra Credit: EPA!

- **Effort**
  - Attending prof and TA office hours, completing all assignments, turning in HW0, doing reading quizzes

- **Participation**
  - Attending lecture and voting using the clickers
  - Asking great questions in discussion and lecture and making it more interactive

- **Altruism**
  - Helping others in lab or on Piazza

- **EPA! extra credit points have the potential to bump students up to the next grade level! (but actual EPA! scores 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% deducted 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

- With the exception of laboratories and assignments that explicitly permit you to work in groups, all homework and projects are to be YOUR work and your work ALONE.
- You are encouraged to discuss your assignments with other students, and extra credit will be assigned to students who help others, particularly by answering questions on Piazza, but 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.
- We have tools and methods, developed over many years, for detecting this. You WILL be caught, and the penalties WILL be severe.
- At the minimum NEGATIVE POINTS for the assignment, probably an F in the course, and a letter to your university record documenting the incidence of cheating.
- (We’ve caught people in recent semesters!)
- Both Giver and Receiver are equally culpable
Architecture of a typical Lecture

- **Full Attention**
  - Administrivia
  - "And in conclusion..."

Time (minutes):
- 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. 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