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")
(lots of help from **TAs,** esp **Head TA Alan Christopher**)
`http://inst.eecs.berkeley.edu/~cs61c/`

Agenda

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

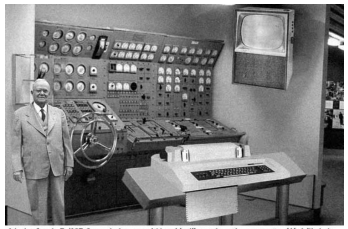
Agenda

- 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

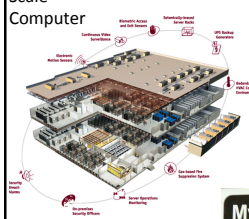
Old School CS61C

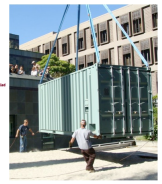


Scientists from the RAND Corporation have created this model to illustrate how a "home computer" could look like in the year 1984. However the needed technology will not be economically feasible for the average home. Also the scientists readily admit that the computer will require not yet invented technology to actually work. But 10 years from now scientific progress is expected to solve these problems. With telepathic interface and the Fortran language, the computer will be easy to use.

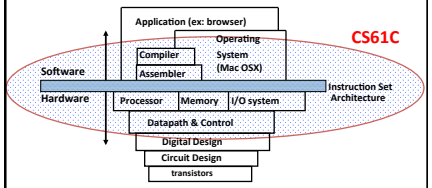
Personal Mobile Devices — New School CS61C (1/2)

Warehouse Scale Computer — New School CS61C (2/2)



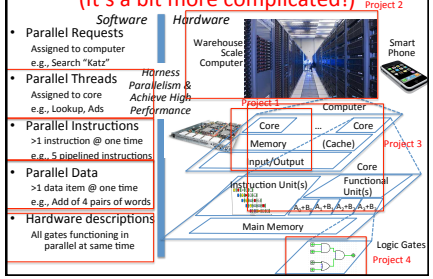


Old School Machine Structures



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
  >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

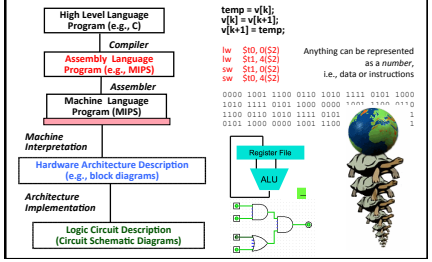
10

6 Great Ideas in Computer Architecture

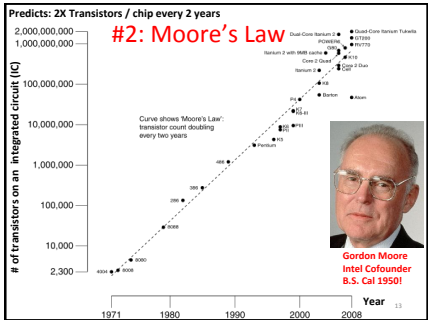
1. Abstraction (Layers of Representation/Interpretation)
2. Moore's Law
3. Principle of Locality/Memory Hierarchy
4. Parallelism
5. Performance Measurement & Improvement
6. Dependability via Redundancy

11

Great Idea #1: Abstraction (Levels of Representation/Interpretation)

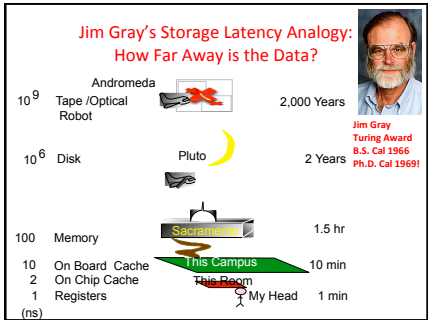


#2: Moore's Law



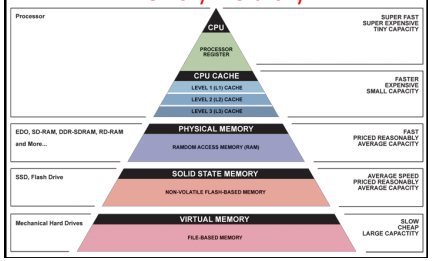
13

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

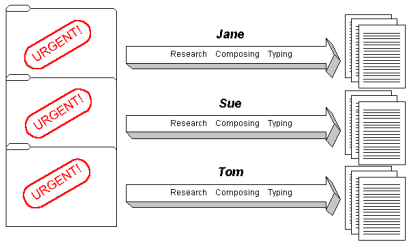


Jim Gray
Turing Award
B.S. Cal 1966
Ph.D. Cal 1969!

Great Idea #3: Principle of Locality/ Memory Hierarchy

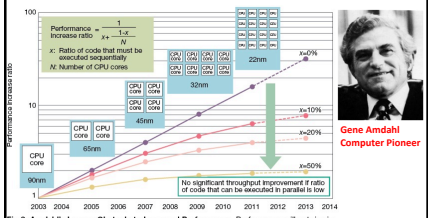


Great Idea #4: Parallelism



16

Caveat: Amdahl's Law



Gene Amdahl
Computer Pioneer

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.

17

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*

18

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

19

Coping with Failures

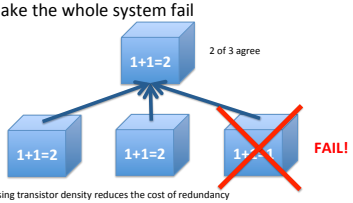
- 4 disks/server, 50,000 servers
- Failure rate of disks: 2% to 10% / year
  - Assume 4% annual failure rate
- On average, how often does a disk fail?
  a) 1 / month
  b) 1 / week
  c) 1 / day
  d) 1 / hour

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

20

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

21

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)


22

Agenda

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

23

Yoda says…

"Always in motion, the future is…"

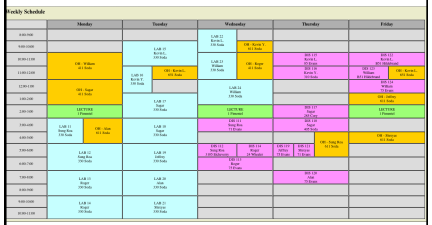
Our schedule may change slightly depending on some factors. This includes lectures, assignments & labs…

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

25

Weekly Schedule



Course Information

- Course Web: http://inst.eecs.Berkeley.edu/~cs61c/
- Instructors:
  - Dan Garcia
- 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)
  - Kernighan & Ritchie, *The C Programming Language*, 2nd Edition
  - Barroso & Holzle, *The Datacenter as a Computer, 1st Edition*
- Piazza:
  - Every announcement, discussion, clarification happens there

27

Course Organization

- 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)
  - Performance Competition for honor (and EPA)
  - Midterm (25%): 2014-03-03, can be clobbered!
  - Final (35%): 2014-05-13 @ 11:30am-2:30pm

28

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 on Friday

EECS Grading Policy

- 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)

	Fall	Spring
2010	2.81	2.81
2009	2.71	2.81
2008	2.95	2.74
2007	2.67	2.76

30

My goal as an instructor

- To make your experience in CS61C as enjoyable & informative as possible
  - Humor, enthusiasm, graphics & technology-in-the-news in lecture
  - Fun, challenging projects & HW
  - Pro-student policies (exam clobbering)
- To maintain Cal & EECS standards of excellence
  - Your projects & exams will be just as rigorous as every year.
- To be an HKN "7.0" man
  - I know I speak fast when I get excited about material. I'm told every semester. Help me slow down when I go toooo fast.
  - Please give me feedback so I improve! Why am I not 7.0 for you? I will listen!!

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!

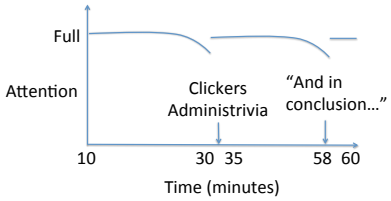
33

Policy on Assignments and Independent Work

- **ALL PROJECTS WILL BE DONE WITH A PARTNER**
- 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.
- PARTNER TEAMS MAY NOT WORK WITH OTHER PARTNER TEAMS
- 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.
- It is NOT acceptable to use PUBLIC github archives (giving your answers away)
- 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 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 and suffer equal penalties**

34

Architecture of a typical Lecture



36

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

37