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

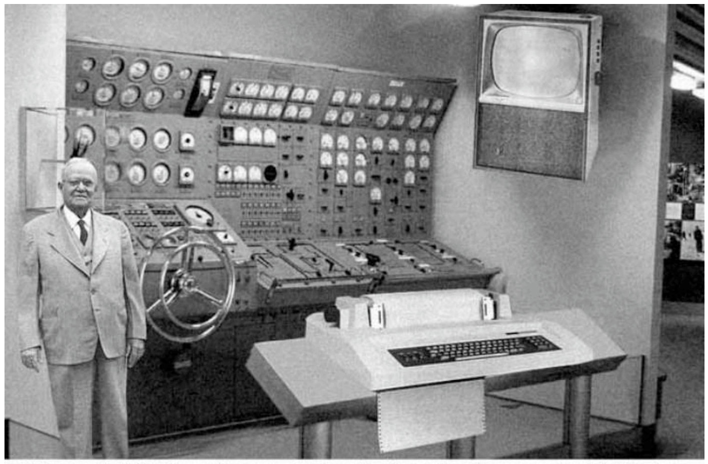
Agenda

- 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



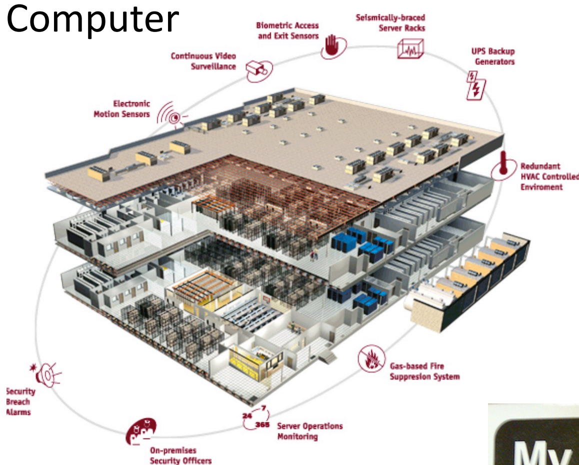
Scientists from the RAND Corporation have created this model to illustrate how a "home computer" could look like in the year 2004. However the needed technology will not be economically feasible for the average home. Also the scientists readily admit that the computer will require not yet invented technology to actually work, but 50 years from now scientific progress is expected to solve these problems. With teletype interface and the Fortran language, the computer will be easy to use.

Personal Mobile Devices

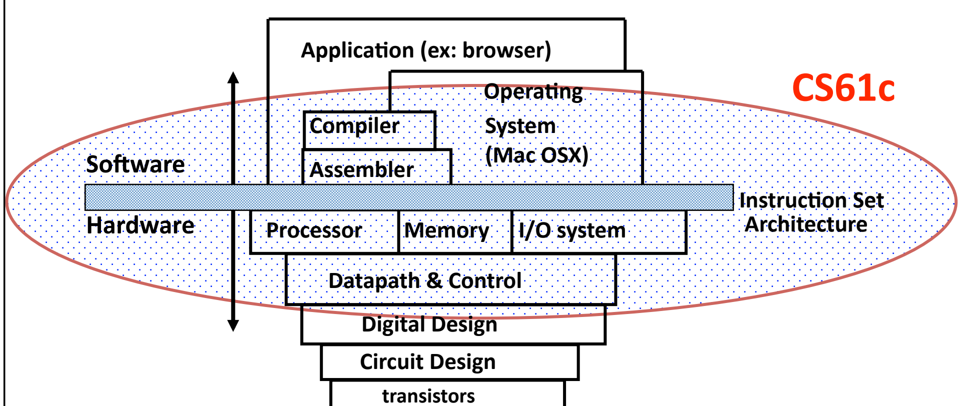
(kinda)New School CS61c (1)

Warehouse Scale Computer

New School CS61c (2)



Old-School Machine Structures



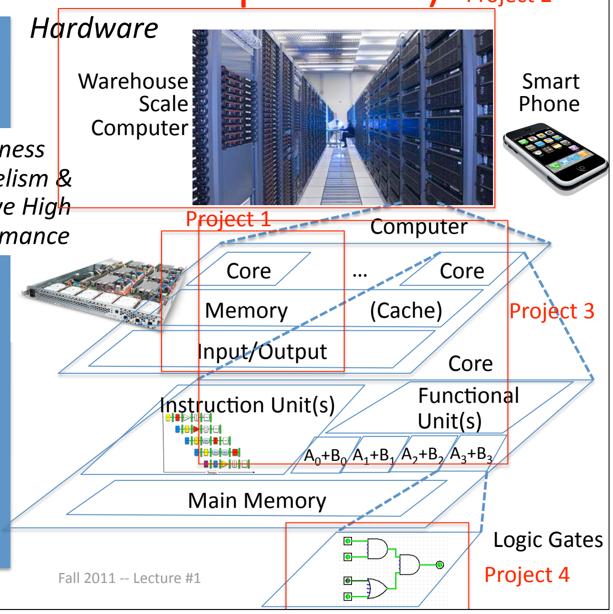
New-School Machine Structures (It's a bit more complicated!)

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., 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

Harness Parallelism & Achieve High Performance

Hardware



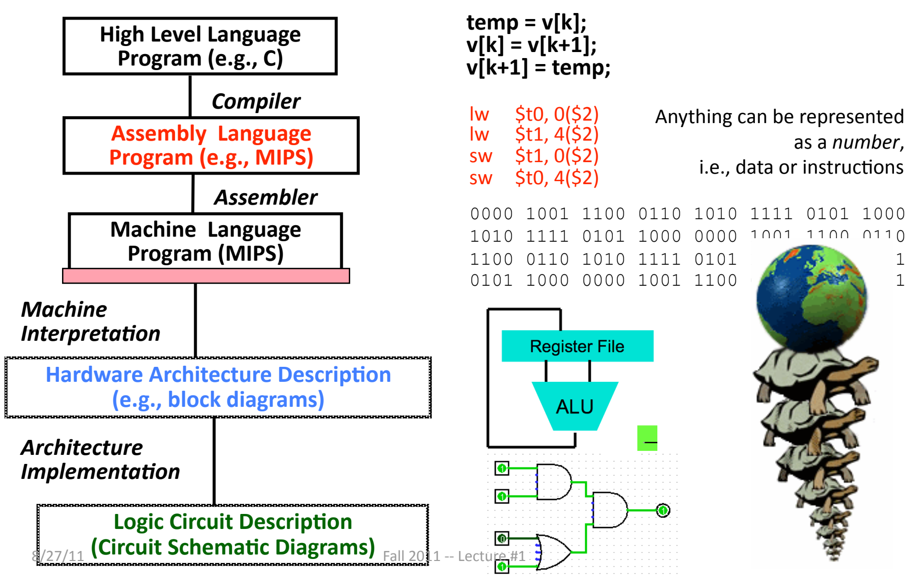
Agenda

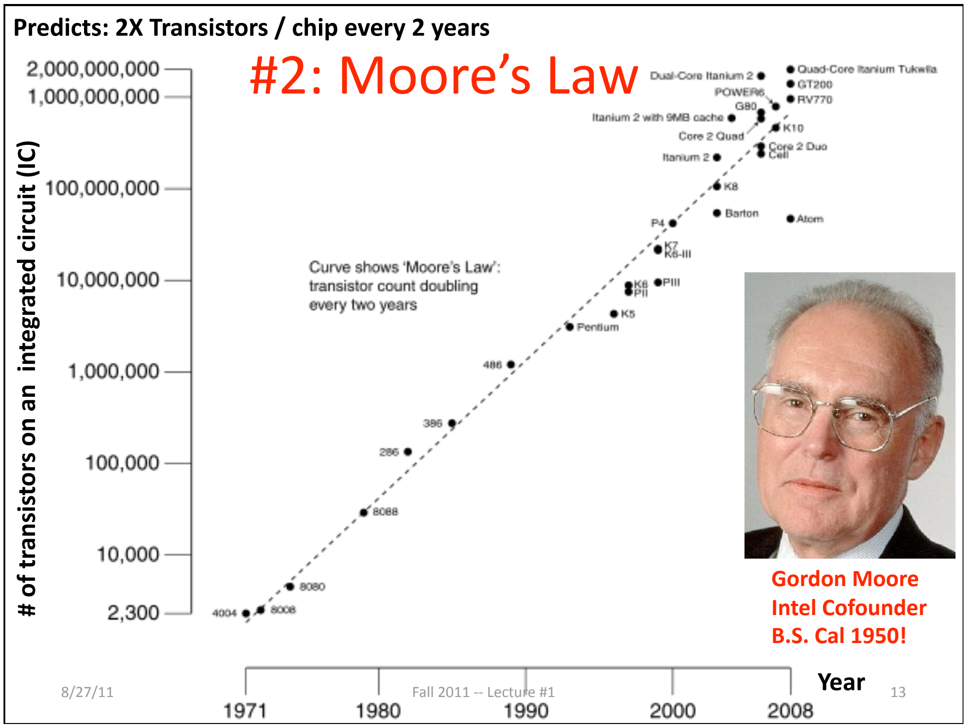
- Thinking about Machine Structures
- Great Ideas in Computer Architecture
- What you need to know about this class

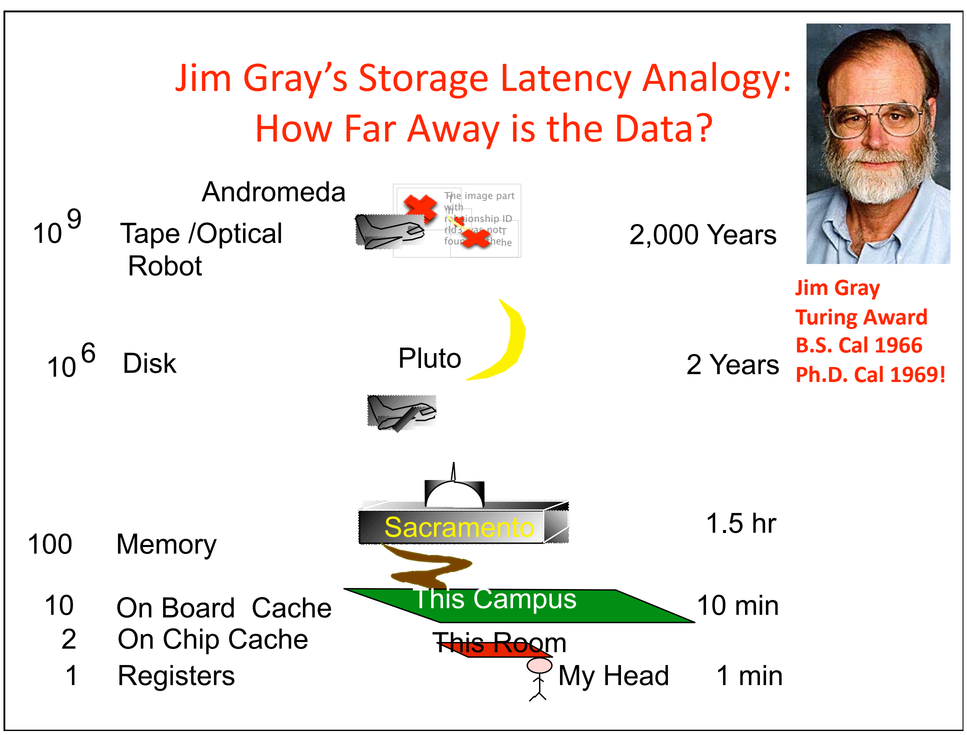
6 Great Ideas in Computer Architecture

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

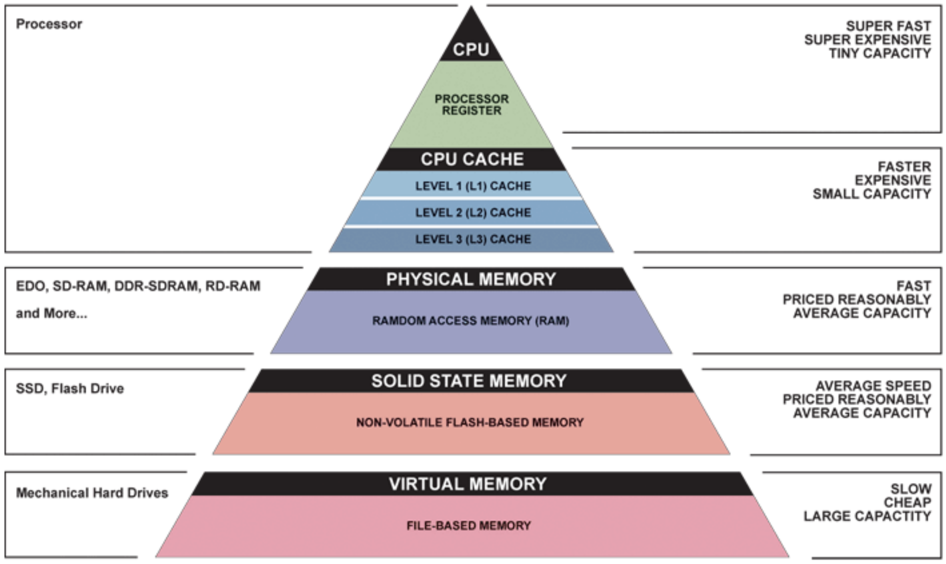
Great Idea #1: Levels of Representation/Interpretation



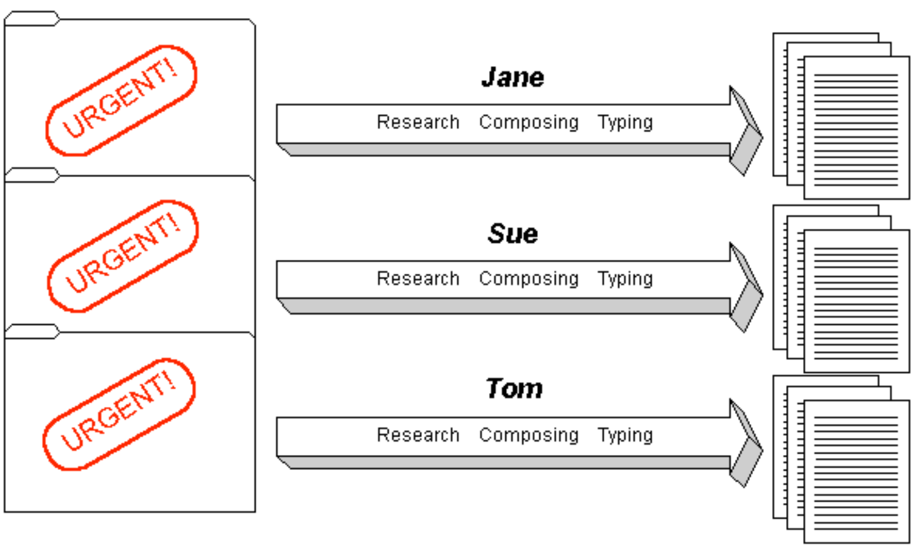

Predicts: 2X Transistors / chip every 2 years
#2: Moore's Law
# of transistors on an integrated circuit (IC)
2,000,000,000
1,000,000,000
100,000,000
10,000,000
1,000,000
100,000
10,000
2,300
Curve shows 'Moore's Law': transistor count doubling every two years
4004
8008
8080
8088
286
386
486
Pentium
K5
K6
PII
PIII
K7
K6-III
P4
Barton
Atom
K8
Itanium 2
Core 2 Quad
Core 2 Duo
Cell
Itanium 2 with 9MB cache
K10
G80
POWER6
Dual-Core Itanium 2
Quad-Core Itanium Tukwila
GT200
RV770
Gordon Moore
Intel Cofounder
B.S. Cal 1950!
8/27/11
Fall 2011 -- Lecture #1
13
Year
1971
1980
1990
2000
2008


Jim Gray's Storage Latency Analogy:
How Far Away is the Data?
Jim Gray
Turing Award
B.S. Cal 1966
Ph.D. Cal 1969!
10 9 Tape /Optical Robot
Andromeda
2,000 Years
10 6 Disk
Pluto
2 Years
100 Memory
Sacramento
1.5 hr
10 On Board Cache
This Campus
10 min
2 On Chip Cache
This Room
1 Registers
My Head
1 min

Great Idea #3: Principle of Locality/ Memory Hierarchy



Great Idea #4: Parallelism



8/27/11 Fall 2011 -- Lecture #1 16

Caveat: Amdahl's Law

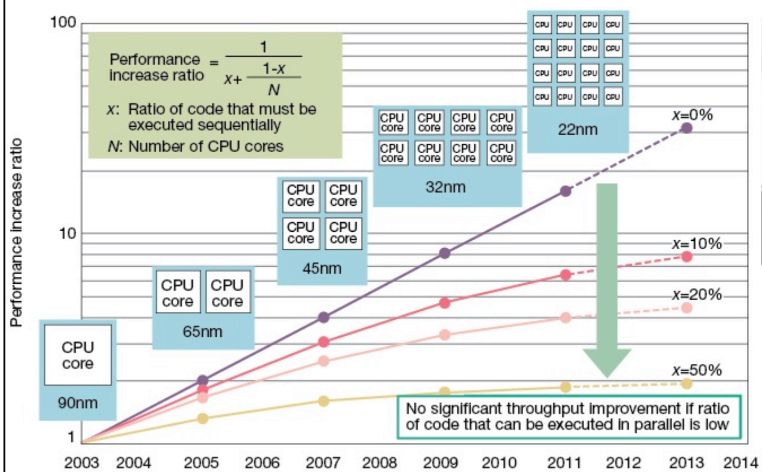


Fig 3 Amdahl's Law an Obstacle to Improved Performance Performance will not rise in the same proportion as the increase in CPU cores. Performance gains are limited by the ratio of software processing that must be executed sequentially. Amdahl's Law is a major obstacle in boosting multicore microprocessor 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.

Gene Amdahl
Computer Pioneer
Ph.D. Wisconsin
1952!

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

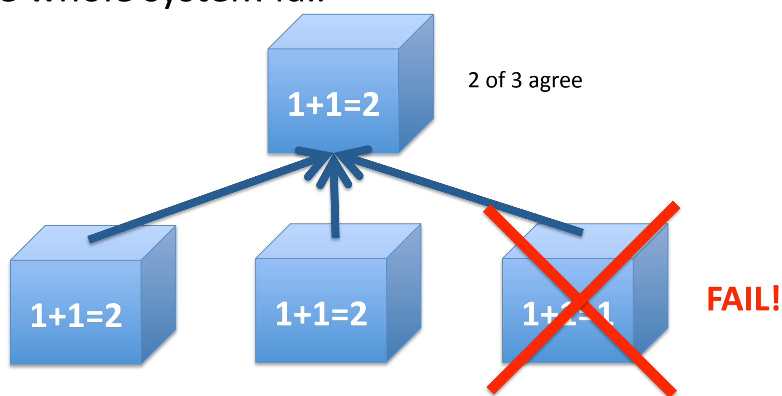
Coping with Failures

- 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 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

Great Idea #6: Dependability via Redundancy

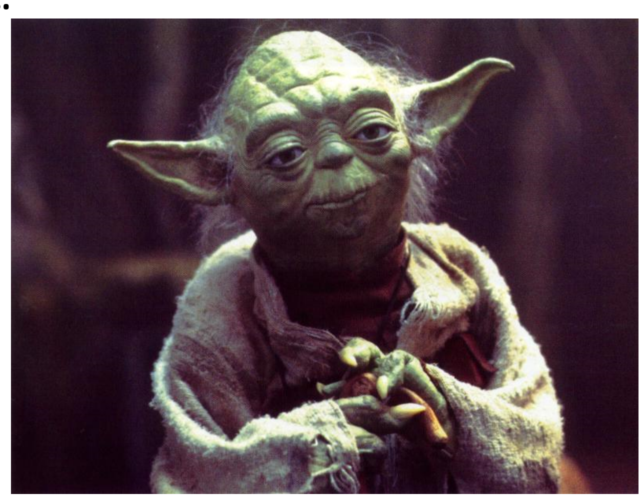
- Applies to everything from datacenters to storage to memory
  - Redundant datacenters so that can lose 1 datacenter but Internet service stays online
  - Redundant disks so that can lose 1 disk but not lose data (Redundant Arrays of Independent Disks/RAID)
  - Redundant memory bits of so that can lose 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…"

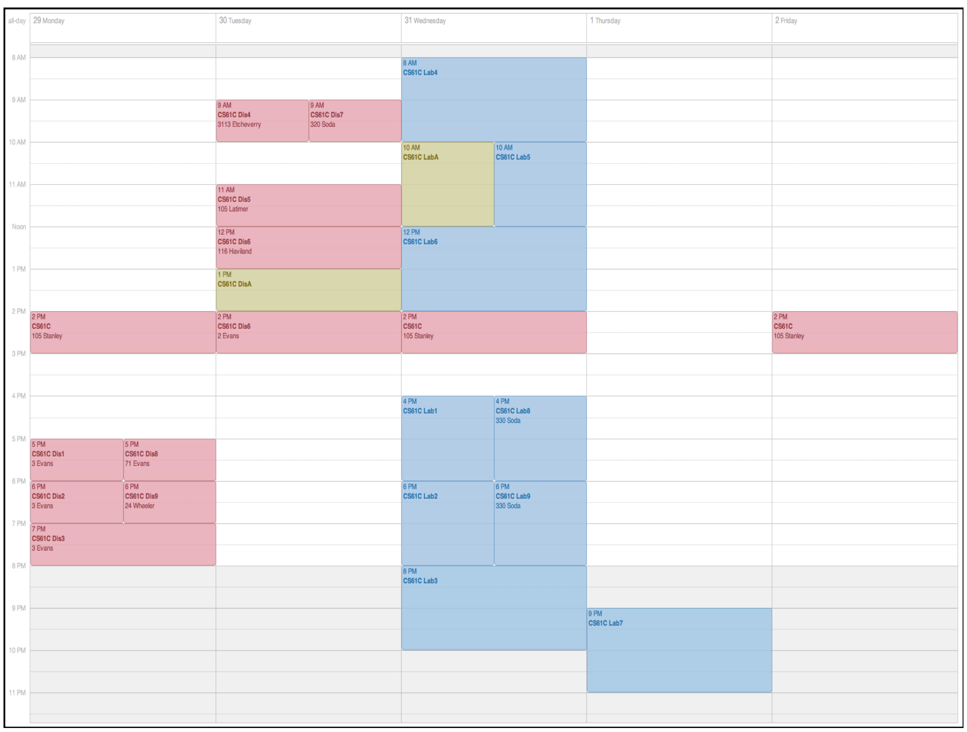
Yoda says…



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

- Course Web: http://inst.eecs.Berkeley.edu/~cs61c/
- 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!)
  - Patterson & Hennessey, *Computer Organization and Design*, 4^{th} Edition (not $\leq 3^{rd}$ Edition, not Asian version 4^{th} edition)
  - Kernighan & Ritchie, *The C Programming Language*, 2^{nd} Edition
  - Barroso & Holzle, *The Datacenter as a Computer, 1^{st} 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
  "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

8/27/11 Fall 2011 -- Lecture #1 31

Extra Credit: EPA!

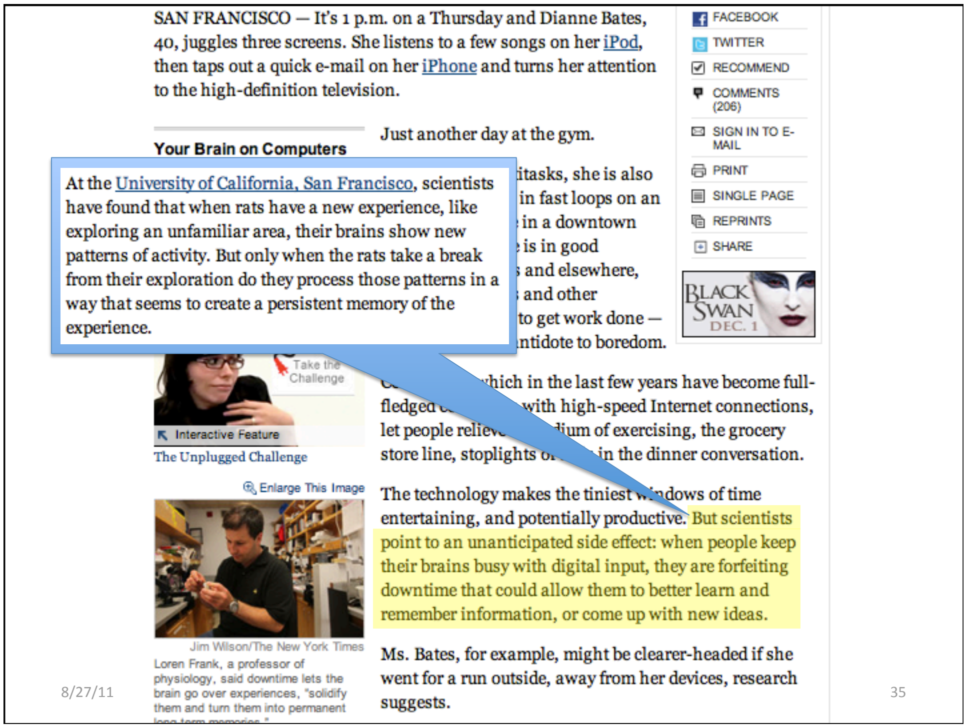
- Effort
  - Attending prof and TA office hours, completing all assignments, turning in HW0, doing reading quizzes
- Participation
  - Attending lecture and voting using the clickers
  - Asking great questions in discussion and lecture and making it more interactive
- Altruism
  - Helping others in lab 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



SAN FRANCISCO — It's 1 p.m. on a Thursday and Dianne Bates, 40, juggles three screens. She listens to a few songs on her iPod, then taps out a quick e-mail on her iPhone and turns her attention to the high-definition television.

Just another day at the gym.

Your Brain on Computers

At the University of California, San Francisco, scientists have found that when rats have a new experience, like exploring an unfamiliar area, their brains show new patterns of activity. But only when the rats take a break from their exploration do they process those patterns in a way that seems to create a persistent memory of the experience.

itasks, she is also in fast loops on an in a downtown is in good and elsewhere, and other to get work done — ntidote to boredom.

hich in the last few years have become full-fledged with high-speed Internet connections, let people relie ium of exercising, the grocery store line, stoplights o in the dinner conversation.

The technology makes the tiniest windows of time entertaining, and potentially productive. But scientists point to an unanticipated side effect: when people keep their brains busy with digital input, they are forfeiting downtime that could allow them to better learn and remember information, or come up with new ideas.

Ms. Bates, for example, might be clearer-headed if she went for a run outside, away from her devices, research suggests.

Take the Challenge

Interactive Feature

The Unplugged Challenge

Enlarge This Image

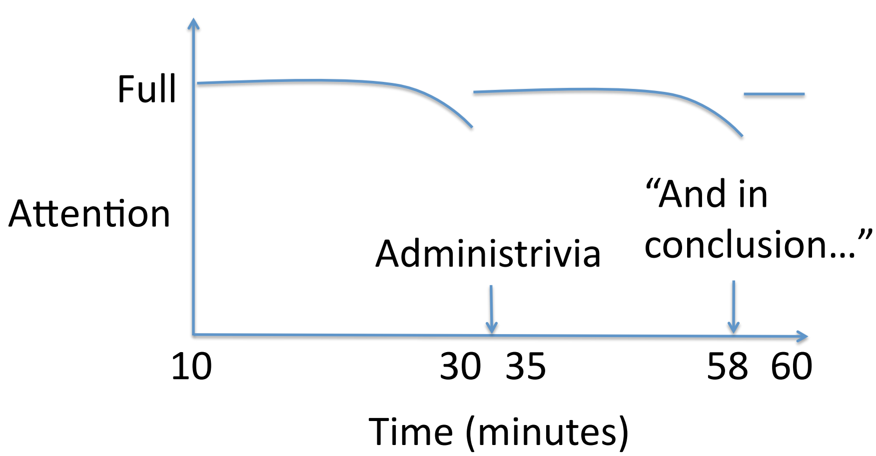
Jim Wilson/The New York Times

Loren Frank, a professor of physiology, said downtime lets the brain go over experiences, "solidify them and turn them into permanent long-term memories."

FACEBOOK
TWITTER
RECOMMEND
COMMENTS (206)
SIGN IN TO E-MAIL
PRINT
SINGLE PAGE
REPRINTS
SHARE

BLACK SWAN DEC. 1

Architecture of a typical Lecture



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