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

Instructors:
Randy H. Katz
David A. Patterson

http://inst.eecs.Berkeley.edu/~cs61c/fa10

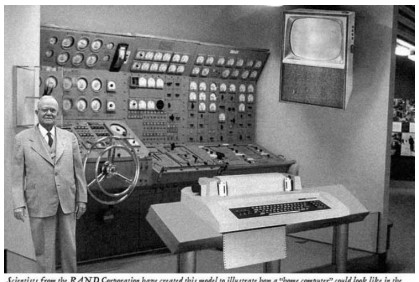
Agenda

- Great Ideas in Computer Architecture
- Administrivia
- Technology Break
- How to Talk to a Computer

Agenda

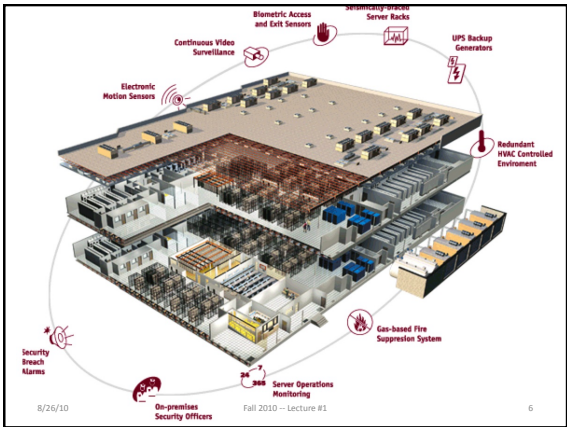
- Great Ideas in Computer Architecture
- Administrivia
- Technology Break
- How to Talk to a Computer

Old School CS61c

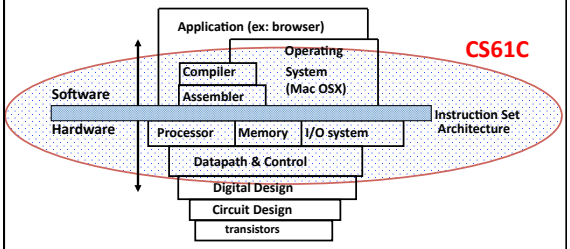


Scientists from the RAND Corporation have created this model to illustrate how a "home computer" could look like in the year 2004. However the needed technology will not be economically feasible for the average home. Also the scientists readily admit that the computer will require not yet invented technology to actually work, but 30 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.

New School CS61c



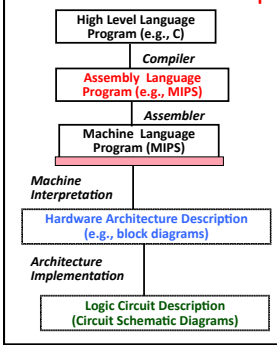
Machine Structures



Great Ideas in Computer Architecture

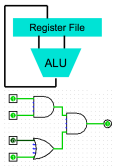
- Layers of Representation/Interpretation
- Moore's Law
- Principle of Locality/Memory Hierarchy
- Parallelism
- Performance Improvement

Levels of Representation/ Interpretation

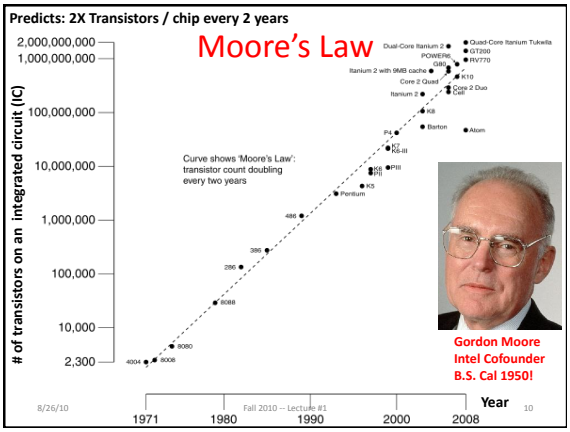



```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;

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

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
```

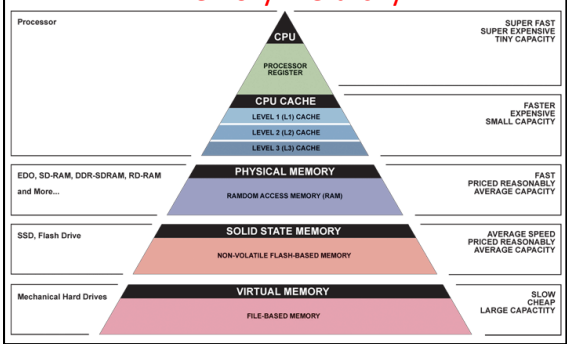
Anything can be represented as a *number*, i.e., data or instructions



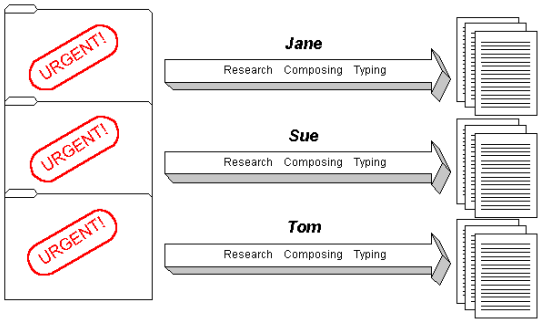
Moore's Law



Principle of Locality/ Memory Hierarchy



Parallelism



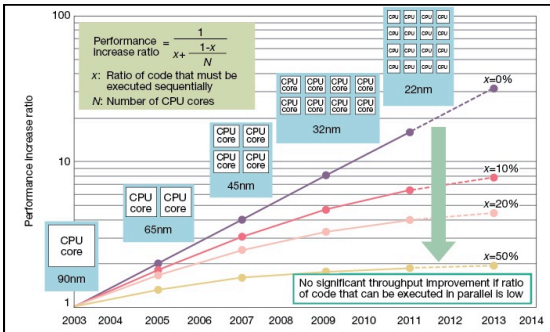


Fig 3 Amdahl's Law an Obstacle to Improved Performance Performance will not rise in the same proportion as the increase in CPU cores. Performance gains are limited by the ratio of software processing that must be executed sequentially. Amdahl's Law is a major obstacle in boosting multicore microprocessor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

Agenda

- Great Ideas in Computer Architecture
- Administrivia
- Technology Break
- How to Talk to a Computer

Course Information

- Instructors:
  - Randy Katz, Dave Patterson
- Teaching Assistants:
  - Andrew Gearhart, Michael Greenbaum, Conor Hughes, Charles Reiss
- Textbooks:
  - Patterson, Hennessey, *Computer Organization and Design*, 4th Edition
  - Kernighan, Ritchie, *The C Programming Language*, 2nd Edition
- Course Web: http://inst.eecs.Berkeley.edu/~cs61c/fa10
- Google Group: 61CFall2010UCB

Course Organization

- Grading
  - Class Participation (5%)
  - Homework (5%)
  - Labs (20%)
  - Projects (40%)
    - Computer Instruction Set Simulator
    - Data Parallelism (Map-Reduce on EC2)
    - Computer Processor Design (Logisim)
    - Performance Tuning of a Parallel Application (partnered)
  - Midterm (10%)
  - Final (20%)

Effort
Participation
Altruism


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.

Interactive Feature

The Unplugged Challenge

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

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.

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

The Rules
(and we really mean it!)


Agenda

- Great Ideas in Computer Architecture
- Administrivia
- Technology Break
- How to Talk to a Computer

Agenda

- Great Ideas in Computer Architecture
- Administrivia
- Technology Break
- How to Talk to a Computer

Speaking to a Computer

- A word a computer understands called an *instruction*
- Vocabulary of all words a computer understands are called an *instruction set*
- Different computers may have different vocabularies (instruction sets)
  - Computer vocabulary used in iPhone is not same as computer vocabulary used in Macbook laptop
- Or the same vocabulary
  - iPhone and iPad computers have same instruction set
- Why not all the same? Why not all different? What might be pros and cons?

Instruction Set in this Class

- MIPS
  - Invented by my co-author John Hennessy
  - Standard instruction set for networking gear
- Elegant example of instruction sets called Reduced Instruction Set Computer (RISC)
  - RISC: Limited vocabulary in number of words and in complexity of those words
- Most popular RISC is ARM (Advanced RISC Machine)
  - 5 billion in 2010, in every cell phone (iPhone, iPad, iPod, …)
- Another popular Instruction Set is used in Macbook and in PCs: Intel 80x86 in Core i3, Core i5, Core i7
  - 1/20th as 80x86s made in 2010 (0.3 billion)

Example MIPS Instructions

- Every computer does arithmetic
- Tell a computer to do addition?

```
add a, b, c
```

  - Add **b** to **c** and put sum into **a**
- Needs 3 operands:
  2 sources + 1 destination for sum
- Always just 1 operation per MIPS instruction
- How write same operation in Java?

Guess the other MIPS instructions

- Subtract **c** from **b** and put difference in **a**?
- Multiply **b** by **c** and put product in **a**?
- Divide **b** by **c** and put quotient in **a**?

Guess the other MIPS instructions

- Java operator &: **c** & **b** and put result in **a**?
- Java operator |: **c** | **b** and put result in **a**?
- Java operator <<: **c** << **b** and put result in **a**?
- Java operator >>: **c** >> **b** and put result in **a**?

Example Instructions

- MIPS instructions are inflexible, rigid:
  - Just 1 arithmetic operation per instruction,
  - Always with 3 operands
- How write this Java operation in MIPS?

```
a = b + c + d + e
```


Comments in MIPS

- Can add comments to MIPS instruction by putting # that continues to end of line of text

```
add a, b, c # b + c is placed in a.
add a, a, d # b + c + d is now in a.
add a, a, e # b + c + d + e is in a.
```


Java to MIPS

- Operators almost identical
- What is MIPS code that performs same as?

```
a = b + c;
d = a - e;
```

- What is MIPS code that performs same as?

```
f = (g + h) - (i + j);
```


Peer Instruction

- For a given function, which programming language likely takes the most lines of code?

I. Scheme
II. Java
III. MIPS instructions

Put the three representations in order

A. Most to least: I, II, III
B. I, III, II
C. II, III, I
D. II, I, III
E. III, II, I
F. III, I, II

Summary

- Computer words called instructions
- Computer vocabulary is instruction set
- MIPS is example instruction set in this class
- Rigid format: 1 operation, 2 source operands, 1 destination
  - add, sub, mult, div, and, or, shl, shr
- Like other "programming languages," has comments to help programmer
- Simple mapping from arithmetic expression in Java to MIPS instrutions