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

Course Introduction, Number Representation

Instructor:
Michael Greenbaum

http://inst.eecs.berkeley.edu/~cs61c/su11

My HW0

- Grew up in the San Fernando Valley ("The Valley"). Moved to nearby Agoura Hills during middle school.
- BS, EECS – UC Berkeley, 2008. Entered PhD program studying AI.
- TA'd 5 times. Cs61c twice.
- Enjoy golfing, playing "roguelike" computer games (nethack, Dungeon Crawl).

- I'm not a professor. Call me Michael.
- Teaching this class will be consuming all my time this summer!

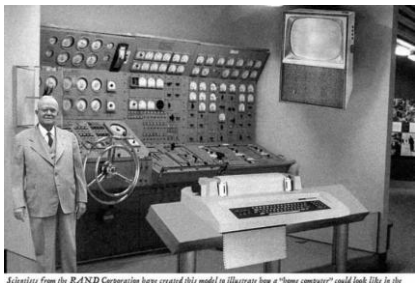
Agenda

- Great Ideas in Computer Architecture
- Administrivia
- Break
- Number Representation

Agenda

- Great Ideas in Computer Architecture
- Administrivia
- Break
- Number Representation

Old School CS61c

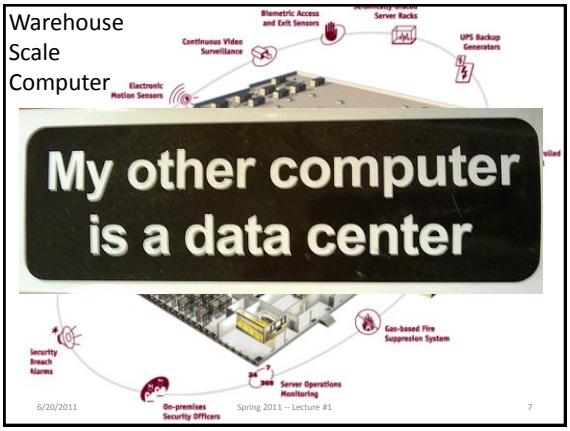


Scientists from the RAND Corporation have created this model to illustrate how a "home computer" could look like in the year 2004. However the needed technology will not be economically feasible for the average home. Also the scientists readily admit that the computer will require not yet invented technology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

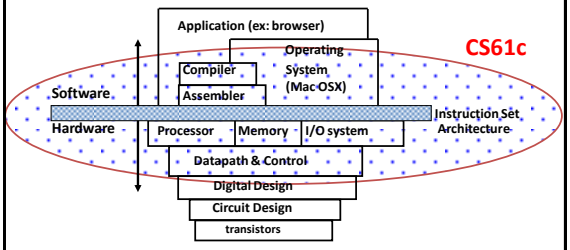
New School CS61c

Personal Mobile Devices

Warehouse Scale Computer



Old-School Machine Structures

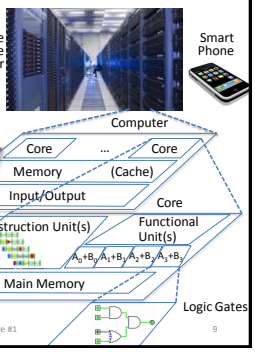


New-School Machine Structures (It's a bit more complicated!)

Software | *Hardware*

- 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

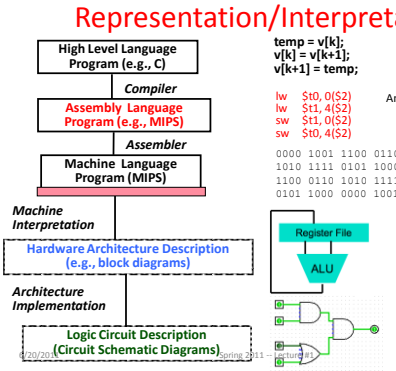
Harness Parallelism & Achieve High Performance



6 Great Ideas in Computer Architecture

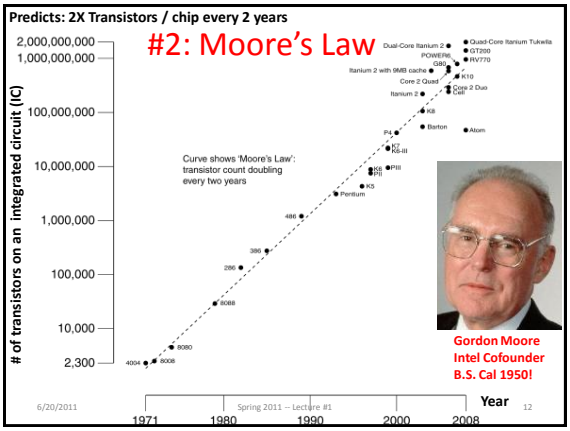
1. Layers of Representation/Interpretation
2. Moore's Law
3. Principle of Locality/Memory Hierarchy
4. Parallelism
5. Performance Measurement & Improvement
6. Dependability via Redundancy

Great Idea #1: Levels of Representation/Interpretation

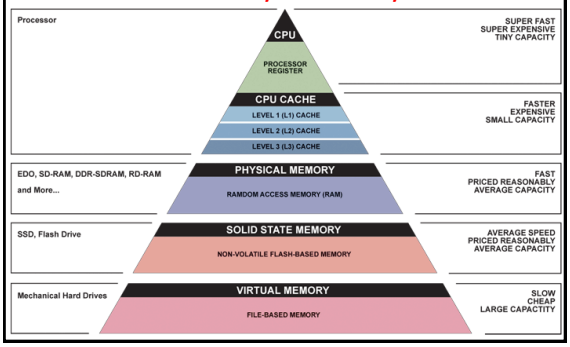


Predicts: 2X Transistors / chip every 2 years

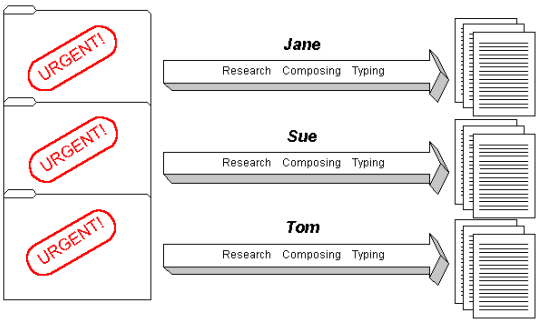
#2: Moore's Law



Great Idea #3: Principle of Locality/ Memory Hierarchy



Great Idea #4: Parallelism



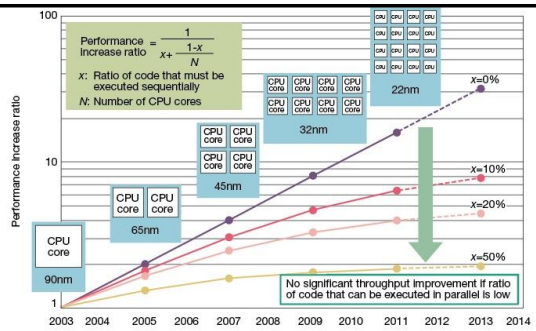
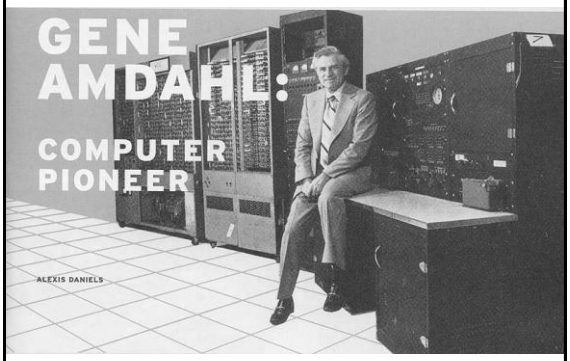


Fig 3 Amdahl's Law an Obstacle to Improved Performance Performance will not rise in the same proportion as the increase in CPU cores. Performance gains are limited by the ratio of software processing that must be executed sequentially. Amdahl's Law is a major obstacle in boosting multicore microprocessor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

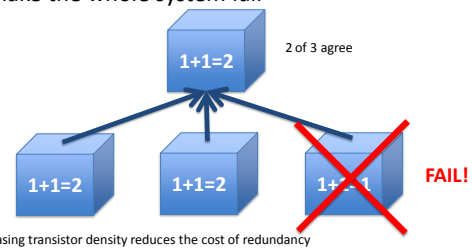


Great Idea #5: Performance Measurement and Improvement

- 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*

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

- Great Ideas in Computer Architecture
- Administrivia
- Break
- Number Representation

Course Information

- Course Website: http://inst.eecs.berkeley.edu/~cs61c/su11
- Textbooks: Average 15 pages of reading/week
  - Patterson & Hennessey, *Computer Organization and Design*, 4th Edition (not ≤3rd Edition, not International 4th edition)
  - Kernighan & Ritchie, *The C Programming Language*, 2nd Edition
  - Barroso & Holzle, *The Datacenter as a Computer, 1st Edition*
- Piazzza for main discussion forum.

Your Course Staff

- TAs
  - Justin Hsia
  - Sean Soleyman
  - Alvin Yuan
- 3 Readers
- ~5 Lab Assistants

Course Organization

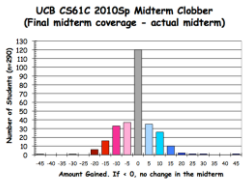
- Grading
  - Effort, Participation, Altruism (EPA) (5%)
  - Homework (5%)
  - Labs (15%)
  - Projects (30%)
    1. Computer Instruction Set Simulator (C)
    2. Performance Tuning of a Parallel Application/Matrix Multiply using cache blocking, SIMD, MIMD (OpenMP, due with partner)
    3. Computer Processor Design (Logisim)
  - Midterm (20%): end of 4th week
  - Final (25%): last day of class

EPA

- Effort
  - Attending Michael's and TA's office hours, completing all assignments, turning in HW0
- Participation
  - Attending lecture and discussion sections.
  - Asking great questions in lecture and discussion and making it more interactive
- Altruism
  - Helping others in lab or on piazzza
- EPA! extra credit points have the potential to bump students up to the next grade level! (but actual EPA! scores are internal)

Exam Clobbering Policy

- A higher score on the "midterm" portion of your final than on your midterm will replace your midterm score!
  - Note: The "midterm" portion of the final still counts toward your score on the final, regardless



Late Policy

- Assignments mostly due Sundays at 11:59:59 PM
- Late homeworks not accepted (100% penalty)
- Late projects get 20% penalty, accepted up to two full days later.
  - No credit if more than 48 hours late
  - No "slip days"
    - Only three projects – Don't want late projects to be the norm.

Policy on Assignments and Independent Work

- With the exception of laboratories and assignments that explicitly permit you to work in groups, all homeworks 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 Piazzza, but we expect that what you hand 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 a ZERO for the assignment, possibly an F in the course, and a letter to your university record documenting the incidence of cheating.
- (We caught people last semester!)

Comments on Summer Version

- Summer is EXTREMELY hectic!
  - Double the standard pace.
- Please stay on top of your reading and assignments
- You won't learn just from lecture. Engaging the material outside of class and lab is critical to your success
- If the course begins to overwhelm you, don't wait, contact me or your TA immediately!

Discussion Sections Moved!

- As part of a Berkeley Institute of Design research project, all sections will be held in 360 Heart Mining Building (at their usual times).
- Your enrollment in CS61C and your class grades will NOT be affected by your decision to participate in the study (sections without full consent will be held in the location without any of the equipment or experiment proceedings)

Other notes

- My OH will be held Monday, Wednesday from 1-2 pm in the Soda Alcoves.

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.

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

BLACK SWAN DEC. 1

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

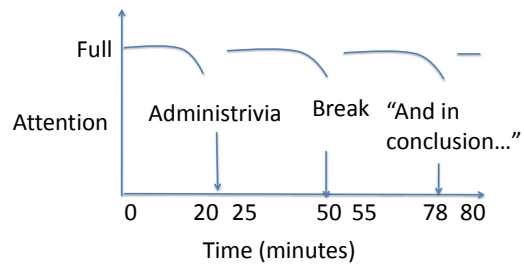
tasks, she is also in fast loops on an in a downtown is in good and elsewhere, and other to get work done — ntidote to boredom.

which in the last few years have become full-fledged with high-speed Internet connections, let people relie dium 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.

Architecture of a Lecture



Agenda

- Great Ideas in Computer Architecture
- Administrivia
- Break
- Number Representation

Number Representation

- Inside a computer, everything stored as a fixed sequence of 0's and 1's.
- How do we represent integers in this format?

Decimal Numbers: Base 10

Digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Example:

9472 = 9000 + 400 + 70 + 2

(9x1000) + (4x100) + (7x10) + (2x1)

$(9 \times 10^3) + (4 \times 10^2) + (7 \times 10^1) + (2 \times 10^0)$

Number Bases

- Number Base B ⇒ B symbols per digit:
  - Base 10 (Decimal): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
    Base 2 (Binary): 0, 1
- Number representation:
  - $d_{n-1}d_{n-2} \ldots d_1d_0$ is an n digit number
  - value = $d_{n-1} \times B^{n-1} + d_{n-2} \times B^{n-2} + \ldots + d_1 \times B^1 + d_0 \times B^0$

Number Bases

- Commonly deal with base 2, base 10, base 16.
  - Will refer to these here as 01101_{two}, 12345_{ten}, and 12345_{hex} for clarity
  - Binary often signified with prefix 0b (eg 0b10101) and hex with 0x.
- An n digit, base B number can represent B^n different things.

Hexadecimal

- Base 16 (hexadecimal) digits: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
- $FFE_{hex} = 15_{ten} \times 16_{ten}^2 + 15_{ten} \times 16_{ten}^1 + 14_{ten} \times 16_{ten}^0$
  $= 3840_{ten} + 240_{ten} + 14_{ten}$
  $= 4094_{ten}$
- Why hex? Because four sequential bits map to one hex digit. Easier to read a long string of binary if converted to hex.
- $1111\ 1111\ 1110_{two} = FFE_{hex} = 4094_{ten}$

Binary	Hex
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

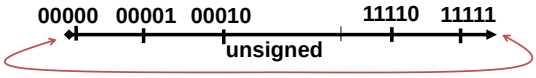
Unsigned Integers

Represent non-negative (unsigned) integers using base 2!

$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_{two} = 0_{ten}$
$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_{two} = 1_{ten}$
$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_{two} = 2_{ten}$
… …
$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101_{two} = 2{,}147{,}483{,}645_{ten}$
$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_{two} = 2{,}147{,}483{,}646_{ten}$
$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_{two} = 2{,}147{,}483{,}647_{ten}$
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_{two} = 2{,}147{,}483{,}648_{ten}$
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_{two} = 2{,}147{,}483{,}649_{ten}$
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_{two} = 2{,}147{,}483{,}650_{ten}$
… …
$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101_{two} = 4{,}294{,}967{,}293_{ten}$
$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_{two} = 4{,}294{,}967{,}294_{ten}$
$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_{two} = 4{,}294{,}967{,}295_{ten}$

What if too big?

- Binary bit patterns above are simply representatives of numbers. Strictly speaking they are called "numerals".
- Numbers really have an ∞ number of digits
  - with almost all being same (00…0) except for a few of the rightmost digits
- If result of add (or -, *, /) cannot be represented by these rightmost hardware bits, overflow is said to have occurred.



Signed Integers

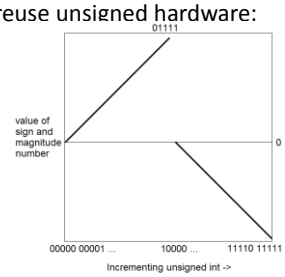
- Programs often need to deal with negative numbers.
- How do we encode these?
- A sequence of n bits can represent 2^n different things.
  - Ideally, want the range evenly split between positive and negative.
- Can we encode them in such a way that we can use the same hardware regardless of whether the numbers are signed or unsigned?

Sign and Magnitude

- Leading (leftmost) bit says whether positive or negative. 0 is positive, 1 is negative.
- Rest of bits treated as usual.
- 8 bit examples:
  - $1000\ 0010 = -2_{ten}$
  - $0000\ 0111 = 7_{ten}$
- What's wrong with this?

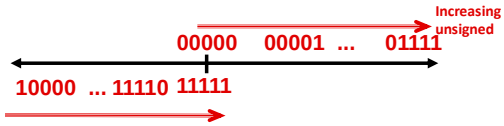
Sign and Magnitude

- Two zeros! 000…0 and 100…0 are +0 and -0.
- Cannot reuse unsigned hardware:



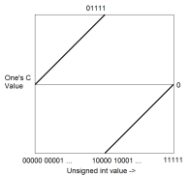
One's Complement

- To negate – complement the bits:
  - +7 => 0000 0111. -7 => 1111 1000.
- Positive numbers always have leading 0's, negative numbers always have leading 1's



One's Complement

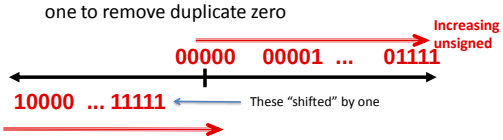
- Incrementing an unsigned int nearly always increments a One's Complement int, with one exception.



- The two zeros: 00…000 and 11…111

Two's Compliment

- Minor modification of one's compliment.
  - "Shift" representation of negative numbers down by one to remove duplicate zero



- Using this representation, incrementing an unsigned int will *always* increment a signed int.

Two's Complement Integers

Sign Bit

$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_{two} = 0_{ten}$
$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_{two} = 1_{ten}$
$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_{two} = 2_{ten}$
… …
$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101_{two} = 2{,}147{,}483{,}645_{ten}$
$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_{two} = 2{,}147{,}483{,}646_{ten}$
$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_{two} = 2{,}147{,}483{,}647_{ten}$
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_{two} = -2{,}147{,}483{,}648_{ten}$
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_{two} = -2{,}147{,}483{,}647_{ten}$
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_{two} = -2{,}147{,}483{,}646_{ten}$
… …
$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101_{two} = -3_{ten}$
$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_{two} = -2_{ten}$
$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_{two} = -1_{ten}$

Two's Complement Summary

- Used by all modern hardware.
- Roughly evenly split between positive and negative.
  - There's one more negative number than positive because the positive side also has zero.
- To negate: Flip the bits and add one:
  - +3 => 0000 0011. Flip bits and add one: 1111 1101 => -3

In Conclusion…

- CS61c: Learn 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 and Improvement
  6. Dependability via Redundancy
- Number Representation – how to represent positive and negative integers using binary.
  - Unsigned – Use straight up base 2.
  - Signed – Two's Complement.