Purpose of today’s lecture is…
….to make this XKCD comic funny!

1...2...
BAAA

...1,306...1,307...
BAAA

...32,767...-32,768...
BAAA

...-32,767...-32,766...
BAAA
“I stand on the shoulders of giants…”

Thanks to these talented folks (& many others) whose contributions have helped make CS61C a really tremendous course!
Where does CS61C fit in?

CS61B No longer a prereq!
Are Computers Smart?

• To a programmer:
  • Very complex operations / functions:
    - `(map (lambda (x) (* x x)) '(1 2 3 4))`
  • Automatic memory management:
    - `List l = new List;`
  • “Basic” structures:
    - Integers, floats, strings, simple commands

Computers are smart!
Are Computers Smart?

• In real life at the lowest level:
  • Only a handful of operations:
    - \{and, or, not\}
  • No automatic memory management.
  • At the lowest level, only 2 values:
    - \{0, 1\} or \{low, high\} or \{off, on\}
What are “Machine Structures”?

Coordination of many levels (layers) of abstraction
CS61C Levels of Representation

High Level Language Program (e.g., C) → Compiler → Assembly Language Program (e.g., MIPS) → Assembler → Machine Language Program (MIPS) → Machine 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)
```

Hardware Architecture Description (e.g., block diagrams) → Architecture Implementation → Logic Circuit Description (Circuit Schematic Diagrams) → Register File → ALU

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

Pearce, Summer 2010 © UCB
Anatomy: 5 components of any Computer

- Processor
  - Control ("brain")
  - Datapath ("brawn")
- Memory
  (where programs, data live when running)
- Devices
  - Input
  - Output
- Keyboard, Mouse
- Disk
  (where programs, data live when not running)
- Display, Printer

John von Neumann invented this architecture
Moore’s Law

Predicts: 2X Transistors / chip every 2 years

Gordon Moore
Intel Cofounder
B.S. Cal 1950!

en.wikipedia.org/wiki/Moore's_law
Technology Trends: Uniprocessor Performance (SPECint)

- **VAX**: 1.25x/year 1978 to 1986
- **RISC + x86**: 1.52x/year 1986 to 2002
- **RISC + x86**: 1.20x/year 2002 to present

"Sea change" in chip design: multiple "cores" or processors per chip
Computer Technology - Growth!

You just learned the difference between (Kilo, Mega, ...) and (Kibi, Mebi, ...).

- **Processor**
  - Speed 2x / 1.5 years (since ’85) [slowing!]
  - 100X performance last decade
  - When you graduate: 4 GHz, 32 Cores

- **Memory (DRAM)**
  - Capacity: 2x / 2 years (since ’96)
  - 64x size last decade.
  - When you graduate: 128 GibiBytes

- **Disk**
  - Capacity: 2x / 1 year (since ’97)
  - 250X size last decade.
  - When you graduate: 8 TeraBytes

...Not nec all on one disk
CS61C: So, what’s in it for me?

- Learn some of the big ideas in CS & Engineering:
  - Principle of abstraction
    - Used to build systems as layers
  - 5 Classic components of a Computer
  - Data can be anything
    - Integers, floating point, characters, ...
    - A program determines what it is
    - Stored program concept: instructions just data
  - Principle of Locality
    - Exploited via a memory hierarchy (cache)
  - Greater performance by exploiting parallelism
  - Compilation v. interpretation through system layers
  - Principles / Pitfalls of Performance Measurement
Others Skills learned in 61C

• Learning C
  • If you know one, you should be able to learn another programming language largely on your own
  • If you know C++ or Java, it should be easy to pick up their ancestor, C

• Assembly Language Programming
  • This is a skill you will pick up, as a side effect of understanding the Big Ideas

• Hardware design
  • We’ll learn just the basics of hardware design
  • CS 150, 152 teach this in more detail
Yoda says…

“Always in motion is the future…”

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

Attention over time!
Attention over time!
Tried-and-True Technique: Peer Instruction

• Increase real-time learning in lecture, test understanding of concepts vs. details

• As we complete a “segment” I’ll ask a 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’ll get transmitters from ASUC bookstore...
Extra Credit: EPA!

- **Effort**
  - Attending Paul’s and TA’s office hours, completing all assignments, turning in HW0

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

- **Altruism**
  - Helping others in lab or on the newsgroup

- **EPA! extra credit points have the potential to bump students up to the next grade level!** (but actual EPA! scores are internal)
Course Problems...Cheating

• What is cheating?
  • **Studying** together in groups is **encouraged**.
  • Common examples of cheating: forging output, taking homework from the box and copy, person asks to borrow solution “just to take a look”, copying a test question...
  • You may discuss the general ideas behind a lab/homework/project problem (except when otherwise noted), but turned-in work must be **completely** your own.
  • You must not ever have in your possession anyone else's solutions (this includes looking at someone else’s code), or give anyone else your solutions
  • **Both “giver” and “receiver” are equally culpable**
  • If you are ever unsure, just ask a TA or Paul.

• **Cheating points:** 0 EPA, negative points for that assignment / project / exam (e.g., if it’s worth 10 pts, you get -10) **In most cases, F in the course.**

• **Every offense will be referred to the Office of Student Judicial Affairs.**
  
  www.eecs.berkeley.edu/Policies/acad.dis.shtml
My goal as an instructor

• To make your experience in CS61C as enjoyable & informative as possible
  • Humor, enthusiasm, and 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. Overall : B- avg

• To be an HKN “7.0” man
  • I know I speak fast when I get excited about material. 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!!
Administrivia

- Upcoming lectures
  - Next three lectures: Introduction to C
  - REALLY REALLY IMPORTANT

- Labs
  - Labs are very overcrowded. We have a 4th lab section now! Consider switching if you can.
  - If you’re checked off in 1st hour, you get an extra point on the labs!

- Enrollment
  - We have room for everyone. If you are on the waitlist, enroll in an open lab!

- Exams are all closed book, can bring 1 sheet of notes front and back

- Soda locks doors @ 6:30pm & on weekends

- Look at class website, newsgroup often!

  http://inst.eecs.berkeley.edu/~cs61c/ucb.class.cs61c
Teaching Assistants

- Noah Johnson (Head TA)
- Eric Chang
- Alex Shyr
- Tom Magrino
- Plus 4 readers and several lab assistants
My HW0

• Grew up outside of LA. Community college transfer to Cal in 07, graduated class of Fall 09.

• Spent last semester as a staff researcher (working on an OS)

• Incoming PhD student, Security and Systems. Currently with the ParLab.

• Please call me Paul. I’m not a professor.

• Avid Bowler

• Programmed in PHP, C, Java, Scheme, x86 ASM, MIPS ASM, Python, et al.

• 2 weeks ago I hiked the Mendenhall glacier and climbed this ice crevasse

• Teaching this class will be consuming all my time this summer!
Class Notes

• Summer is EXTREMELY hectic!
  • ~2 things due per week!

• 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!
Summary

• Continued rapid improvement in computing
  • 2X every 2.0 years in memory size;
    every 1.5 years in processor speed;
    every 1.0 year in disk capacity;
  • Moore’s Law enables processor
    (2X transistors/chip ~1.5-2 yrs)

• 5 classic components of all computers
  Control Datapath Memory Input Output

Processor
Putting it all in perspective…

“If the automobile had followed the same development cycle as the computer,

– Robert X. Cringely
Decimal Numbers: Base 10

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

Example:

$9472 = 9000 + 400 + 70 + 2$

$(9 \times 10^3) + (4 \times 10^2) + (7 \times 10^1) + (2 \times 10^0)$
Numbers: positional notation

- Number Base $B \Rightarrow B$ symbols per digit:
  - Base 10 (Decimal): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
  - Base 2 (Binary): 0, 1

- Number representation:
  - $d_{31}d_{30} \ldots d_1d_0$ is a 32 digit number
  - value = $d_{31} \times B^{31} + d_{30} \times B^{30} + \ldots + d_1 \times B^{1} + d_0 \times B^{0}$

- Binary: 0,1 (In binary digits called “bits”)
  - $0b11010 = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$
  - = 16 + 8 + 2
  - = 26

#s often written

0b… Here 5 digit binary # turns into a 2 digit decimal #

- Can we find a base that converts to binary easily?
  - Yes! Hexadecimal. We’ll cover this in section today.
BIG IDEA: Bits can represent anything!!

- **Characters?**
  - 26 letters ⇒ 5 bits ($2^5 = 32$)
  - upper/lower case + punctuation ⇒ 7 bits (in 8) (“ASCII”)
  - standard code to cover all the world’s languages ⇒ 8,16,32 bits (“Unicode”) 
    www.unicode.com

- **Logical values?**
  - 0 ⇒ False, 1 ⇒ True

- **colors? Ex:**
  - Red (00)
  - Green (01)
  - Blue (11)

- **locations / addresses? commands?**

- **MEMORIZE:** $N$ bits ⇔ at most $2^N$ things
What to do with representations of numbers?

• Just what we do with numbers!
  • Add them
  • Subtract them
  • Multiply them
  • Divide them
  • Compare them

• Example: 10 + 7 = 17
  • …so simple to add in binary that we can build circuits to do it!
  • subtraction just as you would in decimal
  • Comparison: How do you tell if X > Y ?
What if too big?

- Binary bit patterns above are simply representatives of numbers. Strictly speaking they are called “numerals”.

- Numbers really have an $\infty$ number of digits
  - with almost all being same (00…0 or 11…1) except for a few of the rightmost digits
  - Just don’t normally show leading digits

- If result of add (or -, *, /) cannot be represented by these rightmost HW bits, overflow is said to have occurred.
How to Represent Negative Numbers?

(C’s unsigned int, C99’s uintN_t)

- So far, **unsigned numbers**
  - 0 → +
  - 1 → –
  - Rest of bits can be numerical value of number

- **Obvious solution:** define leftmost bit to be sign!

- Representation called **sign and magnitude**

**META:** Ain’t no free lunch
Shortcomings of sign and magnitude?

• Arithmetic circuit complicated
  • Special steps depending whether signs are the same or not

• Also, **two** zeros
  • \(0\times00000000 = +0_{\text{ten}}\)
  • \(0\times80000000 = -0_{\text{ten}}\)
  • What would two 0s mean for programming?

• Also, incrementing “binary odometer”, sometimes increases values, and sometimes decreases!

• Therefore sign and magnitude abandoned
Another try: complement the bits

- Example: \( 7_{10} = 00111_2 \) \( -7_{10} = 11000_2 \)

- Called **One’s Complement**

- Note: positive numbers have leading 0s, negative numbers have leadings 1s.

- What is \(-00000\)? Answer: 11111

- How many positive numbers in N bits?

- How many negative numbers?
Shortcomings of One’s complement?

- Arithmetic still a somewhat complicated.
- Still two zeros
  - \(0x00000000 = +0_{\text{ten}}\)
  - \(0xFFFFFFF = -0_{\text{ten}}\)
- Although used for a while on some computer products, one’s complement was eventually abandoned because another solution was better.
Standard Negative # Representation

• Problem is the negative mappings “overlap” with the positive ones (the two 0s). Want to shift the negative mappings left by one.
  • Solution! For negative numbers, complement, then add 1 to the result

• As with sign and magnitude, & one’s compl. leading 0s ⇒ positive, leading 1s ⇒ negative
  • 000000...xxx is ≥ 0, 111111...xxx is < 0
  • except 1…1111 is -1, not -0 (as in sign & mag.)

• This representation is **Two’s Complement**
  • This makes the hardware simple!

(C’s int, aka a “signed integer”)
(Also C’s short, long long, …, C99’s intN_t)
Two’s Complement Formula

• Can represent positive **and negative** numbers in terms of the bit value times a power of 2:
  
  \[d_{31} \times -(2^{31}) + d_{30} \times 2^{30} + \ldots + d_2 \times 2^2 + d_1 \times 2^1 + d_0 \times 2^0\]

• Example: 1101\textsubscript{two} in a nibble?
  
  \[= 1 \times -(2^3) + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0\]
  
  \[= -2^3 + 2^2 + 0 + 2^0\]
  
  \[= -8 + 4 + 0 + 1\]
  
  \[= -8 + 5\]
  
  \[= -3\text{\,ten}\]

Example: -3 to +3 to -3 (again, in a nibble):

\[
\begin{align*}
\text{x} &: 1101\text{\,two} \\
\text{x'} &: 0010\text{\,two} \\
+1 &: 0011\text{\,two} \\
()' &: 1100\text{\,two} \\
+1 &: 1101\text{\,two}
\end{align*}
\]
2’s Complement Number “line”: $N = 5$

- $2^{N-1}$ non-negatives
- $2^{N-1}$ negatives
- one zero
- how many positives?

Binary odometer

10000 ... 11110 11111
2’s Complement Math

• Lets try adding!
  \(-7 + 3 = -4\)

\[
\begin{array}{c}
1 \\
1 \\
+ \\
+ \\
\hline
1
\end{array}
\]

\[
1 1 0 0 1
\]

\[
+ 0 0 1 1
\]

\[
1 1 0 0
\]

• \(0b1100 = 1 \cdot -(2^3) + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0\)

-8 + 4 = -4

• Or did we just do 9 + 3 = 12? We did both!

• Reuse hardware. Extremely powerful!
And in summary...

- We represent “things” in computers as particular bit patterns: \( N \text{ bits} \Rightarrow 2^N \text{ things} \)

- Different integer encodings have different benefits; 1s complement and sign/mag have most problems.

- **unsigned** (C99’s `uintN_t`):
  
  \[
  \begin{array}{cccccccc}
  00000 & 00001 & \ldots & 01111 & 10000 & \ldots & 11111 \\
  \end{array}
  \]

- 2’s complement (C99’s `intN_t`) universal, learn!

  \[
  \begin{array}{cccccccc}
  00000 & 00001 & \ldots & 01111 \\
  10000 & \ldots & 11110 & 11111 \\
  \end{array}
  \]

- Overflow: numbers \( \infty \); computers finite, errors!

META: We often make design decisions to make HW simple

META: Ain’t no free lunch
REMINDER!

• The class website contains reading assignments that are meant to be done BEFORE each lecture. It is in your best interest to do them!

• It is EXTRA important that you do the reading for next lecture. You will be lost without it!
Reference slides

You ARE responsible for the material on these slides (they’re just taken from the reading anyway) ; we’ve moved them to the end and off-stage to give more breathing room to lecture!
Overview of Physical Implementations

The hardware out of which we make systems.

- Integrated Circuits (ICs)
  - Combinational logic circuits, memory elements, analog interfaces.

- Printed Circuits (PC) boards
  - Substrate for ICs and interconnection, distribution of CLK, Vdd, and GND signals, heat dissipation.

- Power Supplies
  - Converts line AC voltage to regulated DC low voltage levels.

- Chassis (rack, card case, ...)
  - Holds boards, power supply, provides physical interface to user or other systems.

- Connectors and Cables.
**Integrated Circuits (2009 state-of-the-art)**

- Primarily Crystalline Silicon
- 1mm - 25mm on a side
- 2009 feature size $\sim 45 \text{ nm} = 45 \times 10^{-9} \text{ m}$ (then 32, 22, and 16 [by yr 2013])
- 100 - 1000M transistors
- (25 - 100M “logic gates”)
- 3 - 10 conductive layers
- “CMOS” (complementary metal oxide semiconductor) - most common.

**Package provides:**
- spreading of chip-level signal paths to board-level
- heat dissipation.

- Ceramic or plastic with gold wires.
Printed Circuit Boards

- fiberglass or ceramic
- 1-20 conductive layers
- 1-20 in on a side
- IC packages are soldered down.

Provides:
- Mechanical support
- Distribution of power and heat.
Course Lecture Outline

• Basics
  • C-Language, Pointers
  • Memory management

• Machine Representations
  • Numbers (integers, reals)
  • Assembly Programming
  • Compilation, Assembly

• Processors & Hardware
  • Logic Circuit Design
  • CPU organization
  • Pipelining

• Memory Organization
  • Caches
  • Virtual Memory

• I / O
  • Interrupts
  • Disks, Networks

• Advanced Topics
  • Performance
  • Virtualization
  • Parallel Programming
Homeworks, Labs and Projects

- **Lab exercises** (twice a wk; due in that lab session unless extension given by TA) – extra point if you finish in 1st hour!

- **Homework exercises** (~ 2 per week; (HW 0) out now, due in lab TOMORROW)

- **Projects** (~every 2 weeks)

- All exercises, reading, homeworks, projects on course web page

- We will DROP your lowest HW, Lab!
2 Course Exams

• **Midterm:** During 4\textsuperscript{th} week, time TBA
  - Give 3 hours for 2 hour exam
  - Review session beforehand, time/place TBA

• **Final:** Either August 12\textsuperscript{th} or 13\textsuperscript{th}
  - You can *clobber* your midterm grade!
  - (students always LOVE this…)

![Bar graph showing midterm clobber results](image)
Your final grade

• Grading (could change before 1st midterm)
  • 15pts = 5% Labs  
  • 30pts = 10% Homework  
  • 60pts = 20% Projects  
  • 75pts = 25% Midterm* [can be clobbered by Final]  
  • 120pts = 40% Final  
  • + Extra credit for EPA. What’s EPA?

• Grade distributions
  • Similar to CS61[AB], in the absolute scale.  
  • Perfect score is 300 points. 10-20-10 for A+, A, A-  
  • Similar for Bs and Cs (40 pts per letter-grade) 
    … C+, C, C-, D, F (No D+ or D- distinction)  
  • Differs: No F will be given if all-but-one {hw, lab}, all projects submitted and all exams taken  
  • We’ll “ooch” grades up but never down
Texts

• Required: *Computer Organization and Design: The Hardware/Software Interface, Fourth Edition*, Patterson and Hennessy (COD). *The third edition is also accepted.*

• Required: *The C Programming Language*, Kernighan and Ritchie (K&R), 2nd edition

• Reading assignments on web page
Data input: Analog → Digital

• Real world is analog!

• To import analog information, we must do two things
  • Sample
    - E.g., for a CD, every 44,100ths of a second, we ask a music signal how loud it is.
  • Quantize
    - For every one of these samples, we figure out where, on a 16-bit (65,536 tic-mark) "yardstick", it lies.

www.joshuadysart.com/journal/archives/digital_sampling.gif
REFERENCE: Which base do we use?

- **Decimal**: great for humans, especially when doing arithmetic

- **Hex**: if human looking at long strings of binary numbers, its much easier to convert to hex and look 4 bits/symbol
  - Terrible for arithmetic on paper

- **Binary**: what computers use; you will learn how computers do +, -, *, /
  - To a computer, numbers always binary
  - Regardless of how number is written:
    - $32_{\text{ten}} = 32_{10} = 0x20 = 100000_2 = 0b100000$
    - Use subscripts “ten”, “hex”, “two” in book, slides when might be confusing
Bias Encoding: \( N = 5 \) (bias = -15)

- \( \# = \) unsigned + bias
- Bias for \( N \) bits chosen as \(- (2^{N-1} - 1)\)
- one zero
- how many positives?

\[
\begin{array}{c|c|c}
00000 & 00001 & \ldots & 01110 \\
\hline
01111 & 10000 & \ldots & 11110 & 11111 \\
\end{array}
\]
Two’s Complement for N=32

<table>
<thead>
<tr>
<th>Binary (N=32)</th>
<th>Two's Complement</th>
<th>Decimal (N=10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 ... 0000 0000 0000 0000</td>
<td>0\textsubscript{two}</td>
<td>0\textsubscript{ten}</td>
</tr>
<tr>
<td>0000 ... 0000 0000 0000 0001</td>
<td>1\textsubscript{two}</td>
<td>1\textsubscript{ten}</td>
</tr>
<tr>
<td>0000 ... 0000 0000 0000 0010</td>
<td>2\textsubscript{two}</td>
<td>2\textsubscript{ten}</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>0111 ... 1111 1111 1111 1101</td>
<td>2,147,483,645\textsubscript{two}</td>
<td>2,147,483,645\textsubscript{ten}</td>
</tr>
<tr>
<td>0111 ... 1111 1111 1111 1110</td>
<td>2,147,483,646\textsubscript{two}</td>
<td>2,147,483,646\textsubscript{ten}</td>
</tr>
<tr>
<td>0111 ... 1111 1111 1111 1111</td>
<td>2,147,483,647\textsubscript{two}</td>
<td>2,147,483,647\textsubscript{ten}</td>
</tr>
<tr>
<td>1000 ... 0000 0000 0000 0000</td>
<td>-2,147,483,648\textsubscript{two}</td>
<td>-2,147,483,648\textsubscript{ten}</td>
</tr>
<tr>
<td>1000 ... 0000 0000 0000 0001</td>
<td>-2,147,483,647\textsubscript{two}</td>
<td>-2,147,483,647\textsubscript{ten}</td>
</tr>
<tr>
<td>1000 ... 0000 0000 0000 0010</td>
<td>-2,147,483,646\textsubscript{two}</td>
<td>-2,147,483,646\textsubscript{ten}</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1111 ... 1111 1111 1111 1101</td>
<td>-3\textsubscript{two}</td>
<td>-3\textsubscript{ten}</td>
</tr>
<tr>
<td>1111 ... 1111 1111 1111 1110</td>
<td>-2\textsubscript{two}</td>
<td>-2\textsubscript{ten}</td>
</tr>
<tr>
<td>1111 ... 1111 1111 1111 1111</td>
<td>-1\textsubscript{two}</td>
<td>-1\textsubscript{ten}</td>
</tr>
</tbody>
</table>

- One zero; 1st bit called **sign bit**
- 1 “extra” negative: no positive 2,147,483,648\textsubscript{ten}
Two’s comp. shortcut: Sign extension

- Convert 2’s complement number rep. using n bits to more than n bits

- Simply replicate the most significant bit (sign bit) of smaller to fill new bits
  - 2’s comp. positive number has infinite 0s
  - 2’s comp. negative number has infinite 1s
  - Binary representation hides leading bits; sign extension restores some of them

- 16-bit -4\textsubscript{ten} to 32-bit:
  
  \begin{align*}
  &1111 1111 1111 1100_{\text{two}} \\
  &1111 1111 1111 1111 1111 1111 1100_{\text{two}}
  \end{align*}