Course Objective

Bridge the gap between hardware and software
  • How a processor works
  • How a computer is organized

Establish a foundation for building higher-level applications
  • How to understand program performance
  • How to understand where the world is going
Where did it begin?

Electrical Switch
- On/Off
- Binary

Transistor

The first transistor on a workbench at AT&T Bell Labs in 1947
1965

- number of transistors that can be integrated on a die would double every 18 to 24 months (i.e., grow exponentially with time)

Amazingly visionary

- 2300 transistors, 1 MHz clock (Intel 4004) - 1971
- 16 Million transistors (Ultra Sparc III)
- 42 Million transistors, 2 GHz clock (Intel Xeon) – 2001
- 55 Million transistors, 3 GHz, 130nm technology, 250mm$^2$ die (Intel Pentium 4) – 2004
- 290+ Million transistors, 3 GHz (Intel Core 2 Duo) – 2007
- 721 Million transistors, 2 GHz (Nehalem) - 2009
- 1.4 Billion transistors, 3.4 GHz Intel Haswell (Quad core) – 2013
Microprocessor Transistor Counts 1971-2011 & Moore’s Law

curve shows transistor count doubling every two years
Processor Performance Increase

- SUN-4/260
- MIPS M/120
- MIPS M2000
- IBM RS6000
- DEC AXP/500
- DEC Alpha 4/266
- DEC Alpha 21264A/667
- DEC Alpha 21264/600
- DEC Alpha 5/500
- DEC Alpha 5/300
- DEC Alpha 5/300
- DEC Alpha 21264/600
- DEC Alpha 5/300
- DEC Alpha 21264A/667
- IBM POWER 100
- IBM Xeon/2000
- Intel Pentium 4/3000
- Intel Xeon/2000

Performance (SPEC Int)

Year

1987 1989 1991 1993 1995 1997 1999 2001 2003
Moore’s Law

1965

- number of transistors that can be integrated on a die would double every 18 to 24 months (i.e., grow exponentially with time)

Amazingly visionary

- 2300 transistors, 1 MHz clock (Intel 4004) - 1971
- 16 Million transistors (Ultra Sparc III)
- 42 Million transistors, 2 GHz clock (Intel Xeon) – 2001
- 55 Million transistors, 3 GHz, 130nm technology, 250mm$^2$ die (Intel Pentium 4) – 2004
- 290+ Million transistors, 3 GHz (Intel Core 2 Duo) – 2007
- 721 Million transistors, 2 GHz (Nehalem) - 2009
- 1.4 Billion transistors, 3.4 GHz Intel Haswell (Quad core) – 2013
Parallelism

CPU: Central Processing Unit
Then and Now

- The first transistor
  - One workbench at AT&T Bell Labs
  - 1947
  - Bardeen, Brattain, and Shockley

- An Intel Haswell
  - 1.4 billion transistors
  - 177 square millimeters
  - Four processing cores
Then and Now

- **The first transistor**
  - One workbench at AT&T Bell Labs
  - 1947
  - Bardeen, Brattain, and Shockley

- **Galaxy Note 3**
  - 8 processing cores
Parallelism

CPU: Central Processing Unit
GPU: Graphics Processing Unit
GPU-type computation offers higher GFlops

(Source: Sam Naffziger, AMD)
Supercomputers

- **Petaflops** ($10^{15}$)
  - GPUs/multicore/100s-1000s cores

---

**China’s Tianhe-2 Supercomputer Maintains Top Spot on 42nd TOP500 List**

2013-11-18 08:29:48+00:00

MANNHEIM, Germany; BERKELEY, Calif.; and KNOXVILLE, Tenn. — Tianhe-2, a supercomputer developed by China’s National University of Defense Technology, retained its position as the world’s No. 1 system with a performance of 33.86 petaflop/s (quadrillions of calculations per second) on the Linpack benchmark, according to the 42nd edition of the twice-yearly TOP500 list of the world’s most powerful supercomputers. The list was announced Nov. 18 at the SC13 conference in Denver, Colo.

Titan, a Cray XK7 system installed at the Department of Energy’s (DOE) Oak Ridge National Laboratory, remains the No. 2 system. It achieved 17.59 Pflop/s on the Linpack benchmark. Titan is one of the most energy efficient systems on the list consuming a total of 8.21 MW and delivering 2.143 gigaflops/W.

Sequoia, an IBM BlueGene/Q system installed at DOE’s Lawrence Livermore National Laboratory, is again the No. 3 system. It was first delivered in 2011 and achieved 17.17 Plop/s on the Linpack benchmark.
single-threaded free lunch

Welcome to the jungle
- cloud-core
- hetero-core

The free lunch is so over

Exit Moore

1975
1980s
1990s
2000s
2010s
20??

2005
2011
Course Objective

Bridge the gap between hardware and software
  • How a processor works
  • How a computer is organized

Establish a foundation for building higher-level applications
  • How to understand program performance
  • How to understand where the world is going
How class is organized

Instructor: Kavita Bala and Hakim Weatherspoon
(kb@cs.cornell.edu, hweather@cs.cornell.edu)

Lecture:
• Tu/Th 1:25-2:40
• Statler Auditorium

Lab sections:
• Start next week
• Carpenter 104 (Blue room)
• Carpenter 235 (Red room)
• Upson B7

Required Textbooks

Suggested Textbook
Who am I?

Prof. Kavita Bala

• Ugrad: IIT Bombay
• PhD: MIT
• Started in compilers and systems
• Moved to graphics
• Also work on parallel processing in graphics
Who am I?

Prof. Hakim Weatherspoon

- (Hakim means Doctor, wise, or prof. in Arabic)
- Background in Education
  - Undergraduate University of Washington
    - Played Varsity Football
      - Some teammates collectively make $100’s of millions
      - I teach!!!
  - Graduate University of California, Berkeley
    - Some class mates collectively make $100’s of millions
    - I teach!!!
- Background in Operating Systems
  - Peer-to-Peer Storage
    - Antiquity project - Secure wide-area distributed system
    - OceanStore project – Store your data for 1000 years
  - Network overlays
    - Bamboo and Tapestry – Find your data around globe
  - Tiny OS
    - Early adopter in 1999, but ultimately chose P2P direction
Who am I?

Cloud computing/storage

- Optimizing a global network of data centers
Course Staff

Lab/Homework TA’s

- Paul Upchurch <paulu@cs.cornell.edu> (PhD)
- Zhiming Shen <zshen@cs.cornell.edu> (PhD)
- Pu Zhang <pz59@cornell.edu> (PhD)
- Andrew Hirsch <akh95@cornell.edu> (PhD)
- Emma Kilfoyle <efk23@cornell.edu> (MEng)
- Roman Averbukh <raa89@cornell.edu> (MEng)
- Lydia Wang <lw354@cornell.edu> (MEng)
- Favian Contreras <fnc4@cornell.edu>
- Victoria Wu <vw52@cornell.edu>
- Detian Shi <ds629@cornell.edu>
- Maxwell Dergosits <mad293@cornell.edu>
- Jimmy Zhu <jhz22@cornell.edu>
- Antoine Pourchet <app63@cornell.edu>
- Brady Jacobs <bij4@cornell.edu>
- Kristen Tierney <kjt54@cornell.edu>
- Gary Zibrat <gdz4@cornell.edu>
- Naman Agarwal <na298@cornell.edu>
- Sanyukta Inamdar <sri7@cornell.edu>
- Sean Salmon <ss2669@cornell.edu>
- Ari Karo <aak82@cornell.edu>
- Brennan Chu <bc385@cornell.edu>

Administrative Assistant:
- Molly Trufant (mjt264@cs.cornell.edu)
Pre-requisites and scheduling

**CS 2110 is required** (Object-Oriented Programming and Data Structures)

- Must have satisfactorily completed CS 2110
- *Cannot take CS 2110 concurrently with CS 3410*

**CS 3420 (ECE 3140) (Embedded Systems)**

- Take either CS 3410 or CS 3420
  - both satisfy CS and ECE requirements
- *However, Need ENGRD 2300 to take CS 3420*

**CS 3110 (Data Structures and Functional Programming)**

- Not advised to take CS 3110 and 3410 together
Pre-requisites and scheduling

CS 2043 (UNIX Tools and Scripting)
- 2-credit course will greatly help with CS 3410.
- Meets Mon, Wed, Fri at 11:15am-12:05pm in Hollister (HLS) B14
- Class started yesterday and ends March 5th

CS 2022 (Introduction to C) and CS 2024 (C++)
- 1 to 2-credit course will greatly help with CS 3410
- Unfortunately, offered in the fall, not spring
- Instead, we will offer a primer to C during lab sections and include some C questions in homeworks
<table>
<thead>
<tr>
<th>Week</th>
<th>Date (Tue)</th>
<th>Lecture#</th>
<th>Lecture Topic</th>
<th>HW</th>
<th>Prelim Evening</th>
<th>Lab Topic</th>
<th>Lab/Proj</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>23 Jan</td>
<td>K&amp;H</td>
<td>Intro</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>28 Jan</td>
<td>24</td>
<td>Logic &amp; Gates</td>
<td></td>
<td></td>
<td>Logisim</td>
<td>Lab 0: Adder/Logisim intro Handout</td>
</tr>
<tr>
<td></td>
<td></td>
<td>59</td>
<td>Numbers &amp; Arithmetic</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4-Feb</td>
<td>40</td>
<td>K(&amp;out) State &amp; FSMs</td>
<td>HW1: Logic, Gates, Numbers, &amp; Arithmetic</td>
<td>ALU/Design Docs</td>
<td>Lab 1: Labout (design doc due)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>6-Feb</td>
<td>60</td>
<td>18-Out</td>
<td>HW2: FSMs, Memory, CPU, Performance, MIPS</td>
<td>Pro 1: MIPS 1 Handout</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>7-Feb</td>
<td>80</td>
<td>18-Out</td>
<td>Pipelined MIPS</td>
<td>C for Java Programmers</td>
<td>Pro 1: Design Doc Due</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>3-Mar</td>
<td>100</td>
<td>Control Hazards &amp; ISA Variations</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>4-Mar</td>
<td>110</td>
<td>RISC &amp; CISC &amp; Prelim 1 Review</td>
<td>Prelim 1</td>
<td>C lecture 2</td>
<td>C lecture 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>12-Mar</td>
<td>120</td>
<td>Calling Conventions</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>11-Mar</td>
<td>130</td>
<td>Calling Conventions</td>
<td>HW3: Calling Conventions, Exit, Gates</td>
<td>MIPS 2</td>
<td>Pro 2: MIPS 2 Handout</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>18-Mar</td>
<td>150</td>
<td>Linkers &amp; and more calling conventions</td>
<td>Linkers</td>
<td>Intro to UNIX/Linux</td>
<td>Pro 3: Design Doc Due</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16-Mar</td>
<td>160</td>
<td>Cache 1</td>
<td></td>
<td>Sh, gcc, How to tunnel</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>25-Mar</td>
<td>170</td>
<td>Cache 2 &amp;</td>
<td></td>
<td>C lecture 3</td>
<td>C lecture 3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>18-Mar</td>
<td>180</td>
<td>Cache 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1-Apr</td>
<td>190</td>
<td>Spring Break</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>8-Apr</td>
<td>200</td>
<td>Virtual Memory 1</td>
<td>HW4: Virtual memory, Cache, Caches</td>
<td>Stack Smashing</td>
<td>Lab 3: Buffer Overflows handout</td>
<td></td>
</tr>
<tr>
<td></td>
<td>15-Apr</td>
<td>210</td>
<td>Virtual Memory 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>15-Apr</td>
<td>220</td>
<td>Traps</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>22-Apr</td>
<td>230</td>
<td>Multicore Architectures &amp; GPUs</td>
<td>Traps, Multicore, Synchronization</td>
<td>Lab 4: (IN-CLASS) Virtual Memory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>29-Apr</td>
<td>240</td>
<td>Synchronization</td>
<td></td>
<td>Virtual Memory</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>26-Apr</td>
<td>250</td>
<td>Synchronization 2</td>
<td></td>
<td>Lab 2: Design Doc Due</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15.5</td>
<td>6-May</td>
<td>260</td>
<td>GPUs &amp; Prelim 2 Review</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>13-May</td>
<td>270</td>
<td>Future Directions</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>20-May</td>
<td>280</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Note: Spring Break, Winter Break, and Summer Break dates are subject to change.*
Grading

Lab (50% approx.)

• 5-6 Individual Labs
  – 2 out-of-class labs (5-10%)
  – 3-4 in-class labs (5-7.5%)
• 4 Group Projects (30-35%)
• Participation/Quizzes in lab (2.5%)

Lecture (50% approx.)

• 2 Prelims (35%)
  – Dates: March 4, May 1
• Homework (10%)
• Participation/Quizzes in lecture (5%)
Grading

Regrade policy
- Submit written request to lead TA, and lead TA will pick a different grader
- Submit another written request, lead TA will regrade directly
- Submit yet another written request for professor to regrade

Late Policy
- Each person has a total of four “slip days”
- Max of two slip days for any individual assignment
- For projects, slip days are deducted from all partners
- 25% deducted per day late after slip days are exhausted
Active Learning

iClicker: Bring to every Lecture

Put all devices into *Airplane Mode*
Active Learning

Fig. 1 Histogram of 270 physic student scores for the two sections: Experiment w/ quizzes and active learning. Control without.

L Deslauriers et al. Science 2011;332:862-864
Active Learning

Demo: What year are you in school?

a) Freshman
b) Sophomore
c) Junior
d) Senior
e) Other
Active Learning
Also, activity handouts will be available before class
In front of doors before you walk in
Adminstrivia

http://www.cs.cornell.edu/courses/cs3410/2014sp

• Office Hours / Consulting Hours
• Lecture slides, schedule, and Logisim
• CSUG lab access (esp. second half of course)

Lab Sections (start next week)

<table>
<thead>
<tr>
<th>Day</th>
<th>Time</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>2:55 – 4:10pm</td>
<td>Carpenter Hall 104 (Blue Room)</td>
</tr>
<tr>
<td>W</td>
<td>8:40—9:55am</td>
<td>Carpenter Hall 104 (Blue Room)</td>
</tr>
<tr>
<td>W</td>
<td>11:40am – 12:55pm</td>
<td>Carpenter Hall 104 (Blue Room)</td>
</tr>
<tr>
<td>W</td>
<td>3:35 – 4:50pm</td>
<td>Carpenter Hall 104 (Blue Room)</td>
</tr>
<tr>
<td>W</td>
<td>7:30—8:45pm</td>
<td>Carpenter Hall 235 (Red Room)</td>
</tr>
<tr>
<td>R</td>
<td>8:40 – 9:55pm</td>
<td>Carpenter Hall 104 (Blue Room)</td>
</tr>
<tr>
<td>R</td>
<td>11:40 – 12:55pm</td>
<td>Carpenter Hall 104 (Blue Room)</td>
</tr>
<tr>
<td>R</td>
<td>2:55 – 4:10pm</td>
<td>Carpenter Hall 104 (Blue Room)</td>
</tr>
<tr>
<td>F</td>
<td>8:40 – 9:55am</td>
<td>Carpenter Hall 104 (Blue Room)</td>
</tr>
<tr>
<td>F</td>
<td>11:40am – 12:55pm</td>
<td>Upson B7</td>
</tr>
<tr>
<td>F</td>
<td>2:55 – 4:10pm</td>
<td>Carpenter Hall 104 (Blue Room)</td>
</tr>
</tbody>
</table>

• Labs are separate than lecture and homework
• Bring laptop to Labs
• **Next** week: intro to logisim and building an adder
Administrivia
http://www.cs.cornell.edu/courses/cs3410/2014sp

- Office Hours / Consulting Hours
- Lecture slides, schedule, and Logisim
- CSUG lab access (esp. second half of course)

Course Virtual Machine (VM)

- Identical to CSUG Linux machines
- Download and use for labs and projects
- https://confluence.cornell.edu/display/coecis/CSUG+Lab+VM+Information
Communication

Email
- cs3410-staff-l@cs.cornell.edu
- The email alias goes to me and the TAs, not to whole class

Assignments
- CMS: http://cms.csuglab.cornell.edu

Newsgroup
- For students

iClicker
- http://atcsupport.cit.cornell.edu/pollsrvc/
Lab Sections, Projects, and Homeworks

Lab Sections start *next* week

- Intro to logisim and building an adder

Labs Assignments

- Individual
- One week to finish (usually Monday to Monday)

Projects

- two-person teams
- Find partner in same section

Homeworks

- One before each prelim
- Will be released a few weeks ahead of time
- Finish question after covered in lecture
Academic Integrity

All submitted work must be your own
• OK to study together, but do not share soln’s
• Cite your sources

Project groups submit joint work
• Same rules apply to projects at the group level
• Cannot use of someone else’s soln

Closed-book exams, no calculators

• Stressed? Tempted? Lost?
  • Come see us before due date!

Plagiarism in any form will not be tolerated
Why do CS Students Need Transistors?

4th Generation Intel® Core™ Processor Die Map
22nm Tri-Gate 3-D Transistors

Quad core die shown above  |  Transistor count: 1.4 Billion  |  Die size: 177mm²

** Cache is shared across all 4 cores and processor graphics
Why do CS Students Need Transistors?

Functionality and Performance
To be better Computer Scientists and Engineers

- Abstraction: simplifying complexity
- How is a computer system organized? How do I build it?
- How do I program it? How do I change it?
- How does its design/organization effect performance?
Computer System Organization

- Processor
- Memory
- Processor interface
- Disk and USB interfaces
- Graphics
- I/O bus slots

Parts of the Computer System:
- Hard drive
- Processor
- Fan with cover
- Spot for memory DIMMs
- Spot for battery
- Motherboard
- Fan with cover
- DVD drive
Computer System Organization

Computer System = ?

Input +
Output +
Memory +
Datapath +
Control

Registers
CPU

Video
Network
USB
Serial

Keyboard
Mouse

Memory
Disk
Audio
int x = 10;
x = 2 * x + 15;

C

r0 = 0

MIPS

addi r5, r0, 10 ← r5 = r0 + 10
muli r5, r5, 2 ← r5 = r5 * 2
addi r5, r5, 15 ← r5 = r15 + 15

MIPS

op = addi  r0  r5  10
001000000000010100000000000001010
0000000000000010100101000001000000
00100000101001010000000000001111

op = addi  r5  r5  15
Instruction Set Architecture

ISA

• abstract interface between hardware and the lowest level software

• user portion of the instruction set plus the operating system interfaces used by application programmers
A processor executes instructions

- Processor has some internal state in storage elements (registers)

A memory holds instructions and data

- von Neumann architecture: combined inst and data

A bus connects the two

```
01010000
10010100
...  
```

```
01010000
10010100
...  
```
How to Design a Simple Processor

- **Memory**
  - 32
  - 00
- **PC**
  - 2
- **Register File**
  - r5
  - r0
- **Control**
  - 0 5 5 5
- **ALU**
  - 10

- **New PC Calculation**
- **Instructions**
  - 00: `addi r5, r0, 10`
  - 04: `muli r5, r5, 2`
  - 08: `addi r5, r5, 15`
Inside the Processor

AMD Barcelona: 4 processor cores

Figure from Patterson & Hennesssy, Computer Organization and Design, 4th Edition
How to Program the Processor: MIPS R3000 ISA

Instruction Categories

- Load/Store
- Computational
- Jump and Branch
- Floating Point
  - coprocessor
- Memory Management

<table>
<thead>
<tr>
<th>OP</th>
<th>rs</th>
<th>rt</th>
<th>rd</th>
<th>sa</th>
<th>funct</th>
</tr>
</thead>
<tbody>
<tr>
<td>OP</td>
<td>rs</td>
<td>rt</td>
<td></td>
<td></td>
<td>immediate</td>
</tr>
<tr>
<td>OP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>jump target</td>
</tr>
</tbody>
</table>
Overview

Application
Operating System
Compiler Firmware
Memory system Instr. Set Proc. I/O system
Datapath & Control
Digital Design
Circuit Design
Instruction Set Architecture
Applications

Everything these days!

• Phones, cars, televisions, games, computers,...
Applications

- Cell Phones
- PCs
- TVs

Years:
- 1997
- 1999
- 2001
- 2003
- 2005
- 2007

Units (in millions):
- 1993
- 295
- 405
- 502
- 785
- 1,182

Applications:
- Cell Phones
- PCs
- TVs
- Cloud Computing
- NVidia GPU
- Cell Phone
- Berkeley mote
- Cars
Covered in this course

- Application
- Operating System
- Compiler
- Firmware
- Instruction Set Architecture
- Memory system
- I/O system
- Datapath & Control
- Digital Design
- Circuit Design
Why take this course?

- Basic knowledge needed for *all* other areas of CS: operating systems, compilers, ...
- Levels are not independent
  hardware design ↔ software design ↔ performance
- Crossing boundaries is hard but important
  device drivers
- Good design techniques
  abstraction, layering, pipelining, parallel vs. serial, ...
- Understand where the world is going