CS 3410: Computer System Organization and Programming

Prof. Kavita Bala and Prof. Hakim Weatherspoon
CS 3410, Spring 2014
Computer Science
Cornell University
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
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) - 1971
• 42 Million transistors, 2 GHz clock (Intel Xeon) – 2001
• 55 Million transistors, 3 GHz, 130nm technology, 250mm² 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

The curve shows transistor count doubling every two years.
Processor Performance Increase
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
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

[Image of the first transistor and a diagram of a 4th Generation Intel® Core™ Processor Die Map.](http://techguru3d.com/4th-gen-intel-haswell-processors-architecture-and-lineup/)
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
GPU-type computation offers higher GFlops

![GFLOPs Trend](image)

(Source: Sam Naffziger, AMD)
GPUs: Faster than Moore’s Law

Graph courtesy of Professor John Poulton (from Eric Haines)
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 Pflop/s on the Linpack benchmark.
Why?

- Parallelism
- Pipelining
Programmable Hardware

• Started in 1999
• Flexible, programmable
  – Vertex, Geometry, Fragment Shaders
• And much faster, of course
  • 1999 GeForce256: 0.35 Gigapixel peak fill rate
  • 2001 GeForce3: 0.8 Gigapixel peak fill rate
  • 2003 GeForceFX Ultra: 2.0 Gigapixel peak fill rate
  • ATI Radeon 9800 Pro: 3.0 Gigapixel peak fill rate
  • 2006 NV60: ... Gigapixel peak fill rate
  • 2009 GeForce GTX 285: 10 Gigapixel peak fill rate
  • 2011
    – GeForce GTC 590: 56 Gigapixel peak fill rate
    – Radeon HD 6990: 2x26.5
  • 2012
    – GeForce GTC 690: 62 Gigapixel/s peak fill rate
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
- Computer Organization and Design
- A Reference Manual
- Linux Pocket Guide

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
Autodesk 360 Cloud Render
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 classmates 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

cs3410-staff-l@cs.cornell.edu

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>1K&amp;H</td>
<td>Intro</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>28 Jan</td>
<td>2K</td>
<td>Logic &amp; Gates</td>
<td></td>
<td></td>
<td>Logisim</td>
<td>Lab 0: Adder/Logisim intro Handout</td>
</tr>
<tr>
<td>3</td>
<td>4-Feb</td>
<td>4H</td>
<td>Numbers &amp; Arithmetic</td>
<td>4H1: Logic, Gates, Numbers, &amp; Arithmetic</td>
<td>ALU/Design Docs</td>
<td>Lab 1: ALU Handout (design doc due)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>11-Feb</td>
<td>6K</td>
<td>Simple CPU</td>
<td>6K</td>
<td></td>
<td>Lab 2: (IN-CLASS) FSM Handout</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>18-Feb</td>
<td>6K(out)</td>
<td>Winter Break</td>
<td>6K</td>
<td></td>
<td>Lab 3: MIPS 1 Handout</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>25-Feb</td>
<td>9K</td>
<td>Pipeline Hazards</td>
<td>9K</td>
<td></td>
<td>Lab 4: MIPS 2 Handout</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>4-Mar</td>
<td>11K</td>
<td>Calling Conventions</td>
<td>11K</td>
<td></td>
<td>Lab 5: MIPS 3 Handout</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>11-Mar</td>
<td>13K</td>
<td>Calling Conventions</td>
<td>13K</td>
<td></td>
<td>Lab 6: MIPS 4 Handout</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>18-Mar</td>
<td>15K</td>
<td>Linkers</td>
<td>15K</td>
<td></td>
<td>Lab 7: MIPS 5 Handout</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>25-Mar</td>
<td>17K</td>
<td>Caches 1</td>
<td>17K</td>
<td></td>
<td>Lab 8: MIPS 6 Handout</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1-Apr</td>
<td>19K</td>
<td>Caches 2</td>
<td>19K</td>
<td></td>
<td>Lab 9: MIPS 7 Handout</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>8-Apr</td>
<td>21K</td>
<td>Caches 3</td>
<td>21K</td>
<td></td>
<td>Lab 10: MIPS 8 Handout</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>15-Apr</td>
<td>23K</td>
<td>Traps</td>
<td>23K</td>
<td></td>
<td>Lab 11: MIPS 9 Handout</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>22-Apr</td>
<td>25K</td>
<td>Synchronization</td>
<td>25K</td>
<td></td>
<td>Lab 12: MIPS 10 Handout</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>29-Apr</td>
<td>27K</td>
<td>Synchronization</td>
<td>27K</td>
<td></td>
<td>Lab 13: MIPS 11 Handout</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>5-May</td>
<td>29K</td>
<td>I/O</td>
<td>29K</td>
<td></td>
<td>Lab 14: MIPS 12 Handout</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>12-May</td>
<td>31K</td>
<td>Future Directions</td>
<td>31K</td>
<td></td>
<td>Lab 15: MIPS 13 Handout</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>19-May</td>
<td>33K</td>
<td></td>
<td>33K</td>
<td></td>
<td>Lab 16: MIPS 14 Handout</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>26-May</td>
<td>35K</td>
<td></td>
<td>35K</td>
<td></td>
<td>Lab 17: MIPS 15 Handout</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>3-Jun</td>
<td>37K</td>
<td></td>
<td>37K</td>
<td></td>
<td>Lab 18: MIPS 16 Handout</td>
<td></td>
</tr>
</tbody>
</table>

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

## Lab Sections (start **next week**)

<table>
<thead>
<tr>
<th>Day</th>
<th>Time</th>
<th>Location</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?
Why do CS Students Need Transistors?

*Functionality and Performance*
Why do CS Students Need Transistors?

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
Computer System Organization

Computer System = ?
Input +
Output +
Memory +
Datapath +
Control

CPU

Registers

Video

Network

USB

Serial

Keyboard

Mouse

Memory

Disk

Audio

bus

bus
Compilers & Assemblers

C

int x = 10;
x = 2 * x + 15;

MIPS assembly language

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

MIPS machine language

<table>
<thead>
<tr>
<th>op</th>
<th>r0</th>
<th>r5</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00100000000001010</td>
<td>000000000000010101000010000000</td>
<td>001000000000010101000010000000</td>
<td></td>
</tr>
<tr>
<td>addi</td>
<td>r5</td>
<td>r5</td>
<td>15</td>
</tr>
</tbody>
</table>

r0 = 0
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
Basic Computer System

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
How to Design a Simple Processor

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 & Hennessy, Computer Organization and Design, 4th Edition
How to Program the Processor: 
MIPS R3000 ISA

Instruction Categories

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

Registers

<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>
Applications

Everything these days!

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

- Cell Phones
- PCs
- TVs

<table>
<thead>
<tr>
<th>Year</th>
<th>Cell Phones</th>
<th>PCs</th>
<th>TVs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1997</td>
<td>1193</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1999</td>
<td>295</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2001</td>
<td>135</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2003</td>
<td>502</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2005</td>
<td>785</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2007</td>
<td>1182</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Cloud Computing
- NVidia GPU
- Xilinx FPGA
- Berkeley mote
- Cell Phone
- Cars
Covered in this course

- Application
- Operating System
  - Compiler
  - Firmware
- Instruction Set Architecture
- Instruction Set Proc.
- I/O system
- Memory 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