CS 3410: Computer System Organization and Programming

Prof. Hakim Weatherspoon
CS 3410, Spring 2015
Computer Science
Cornell University
“Sometimes it is the people that no one imagines anything of who do the things that no one can imagine”

--quote from the movie The Imitation Game
“Can machines think?”

-- Alan Turing, 1950

Computing Machinery and Intelligence
Enigma machine
Used by the Germans during World War II to encrypt and exchange secret messages

The Bombe
used by the Allies to break the German Enigma machine during World War II
Turing Machine
1936

Alan Turing
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)
• 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 graph shows a logarithmic relationship between the date of introduction of microprocessors and their transistor count. The curve indicates that the transistor count doubles every two years, illustrating Moore’s Law.

Key points:
- The graph starts with the RCA 1802 in 1971 with a transistor count of 2,300.
- By 2011, the 16-Core SPARC T3 has a transistor count of over 2,600,000,000.
- Key developments include the introduction of the Pentium and Pentium II in the late 1980s and early 1990s, respectively.

The growth in transistor counts is exponential, reflecting advancements in semiconductor technology and manufacturing processes.
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

![Peak GFLOPs Trend](image)

(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.
The diagram illustrates the progression of computing technologies from the 1970s to the present day. The timeline is divided into decades:

- **1970s**
- **1980s**
- **1990s**
- **2000s**
- **2010s**
- **20??** (Future perspective)

The timeline highlights the following periods:

- **Single-threaded free lunch**
- **Multicore**
- **Welcome to the jungle**
  - **Cloud-core**
  - **Hetero-core**

Additional notes include:

- "The free lunch is so over"
- "Exit Moore"
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: Hakim Weatherspoon
(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

Suggested Textbook
Who am I?

Prof. Hakim Weatherspoon
• (Hakim means Doctor, wise, or prof. in Arabic)

Career Path
• Undergrad → grad → post-doc → professor
• Washington → Berkeley → Cornell → Cornell
The promise of the Cloud

- ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

NIST Cloud Definition
Who am I?

The promise of the Cloud

- ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

NIST Cloud Definition
The promise of the Cloud

- ubiquitous, convenient, on-demand access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

NIST Cloud Definition

Who am I?

Requires fundamentals in distributed systems

- Networking
- Computation
- Storage
Who am I?

Cloud computing/storage

- Optimizing a global network of data centers
Course Staff

cs-3410-staff-l@cornell.edu

Lab/Homework TA's
- Deniz Altinbuken <deniz@cs.cornell.edu> (PhD)
- Adam Campbell <atc89@cornell.edu> (PhD)
- Praveen Kumar <praveenk@cs.cornell.edu> (PhD)
- Vishal Shrivastav <vishal@cs.cornell.edu> (PhD)
- Rob Mcguinness <jrm369@cornell.edu> (MEng)
- Akshay Navalakha <adn47@cornell.edu> (MEng)
- Andrew Weymouth <asw75@cornell.edu>
- Naman Agarwal <na298@cornell.edu>
- Maxwell Dergosits <mad293@cornell.edu>
- Antoine Pourchet <app63@cornell.edu>
- Gary Zibrat <gdz4@cornell.edu>
- Ari Karo <aak82@cornell.edu>
- Amy Chen <yc624@cornell.edu>
- Stephanie Guo <lg399@cornell.edu>
- Kylar Henderson <kdh59@cornell.edu>
- Megan Carpenter <mnc29@cornell.edu>
- Rebecca Stambler <rls462@cornell.edu>
- Yogisha Dixit <yad4@cornell.edu>
- Mahak Goel <mg785@cornell.edu>
- Spandan Agrawal <sga35@cornell.edu>
- Adwit Tumuluri <at627@cornell.edu>
- Daniel Liu <dl596@cornell.edu>
- Rishab Gupta <rsg246@cornell.edu>
- Oscar Pacheco <ofp3@cornell.edu>
- Lucas Derraugh <ldd49@cornell.edu>
- Brian Wang <bhw45@cornell.edu>
- Anthony Lin <al744@cornell.edu>
- Charles Lai <cjil223@cornell.edu>
- Jonathan Behrens <jkb229@cornell.edu>

Administrative Assistant:
- Jessica Depew <jd648@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>22-Jan</td>
<td>1</td>
<td>Intro</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>27-Jan</td>
<td>2</td>
<td>Logic &amp; Gates</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3-Feb</td>
<td>3</td>
<td>Numbers &amp; Arithmetic</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4-Feb</td>
<td>4</td>
<td>State &amp; FSMs</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>10-Feb</td>
<td>5</td>
<td>Memory</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>17-Feb</td>
<td>6</td>
<td>Simple CPU</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>24-Feb</td>
<td>7</td>
<td>CPU Performance &amp; Pipelines</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>3-Mar</td>
<td>8</td>
<td>Winter Break</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>10-Mar</td>
<td>9</td>
<td>Pipelined MIPS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>17-Mar</td>
<td>10</td>
<td>Pipeline Hazards</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>3-Mar</td>
<td>11</td>
<td>RISC &amp; CISC &amp; Prelim 1 Review</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>10-Mar</td>
<td>12</td>
<td>Calling Conventions</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>17-Mar</td>
<td>13</td>
<td>Calling Conventions</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>24-Mar</td>
<td>14</td>
<td>Calling Conventions</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>7-Apr</td>
<td>15</td>
<td>Linkers</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>14-Apr</td>
<td>16</td>
<td>Caches 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>21-Apr</td>
<td>17</td>
<td>Caches 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>28-Apr</td>
<td>18</td>
<td>Caches 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>5-May</td>
<td>19</td>
<td>Virtual Memory 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>12-May</td>
<td>20</td>
<td>Virtual Memory 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>19-May</td>
<td>21</td>
<td>Virtual Memory 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>26-May</td>
<td>22</td>
<td>Virtual Memory 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>3-Mar</td>
<td>23</td>
<td>Virtual Memory 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>10-Mar</td>
<td>24</td>
<td>Virtual Memory 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>17-Mar</td>
<td>25</td>
<td>Virtual Memory 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>24-Mar</td>
<td>26</td>
<td>Virtual Memory 8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>7-Apr</td>
<td>27</td>
<td>Virtual Memory 9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>14-Apr</td>
<td>28</td>
<td>Virtual Memory 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>21-Apr</td>
<td>29</td>
<td>Virtual Memory 11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>28-Apr</td>
<td>30</td>
<td>Virtual Memory 12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>5-May</td>
<td>31</td>
<td>Virtual Memory 13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>12-May</td>
<td>32</td>
<td>Virtual Memory 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>19-May</td>
<td>33</td>
<td>Virtual Memory 15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 3, April 30
• 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/cs3410/2015sp
• 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>1:25—2:40pm</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 (Blue 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>Carpenter Hall 104 (Blue Room)</td>
</tr>
<tr>
<td>F</td>
<td>1:25 – 2:40pm</td>
<td>Carpenter Hall 104 (Blue Room)</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
• This week: “hello world” lab: Intro to C and virtual machines
• Next week: Intro to logisim, logic circuits, and building an adder
Administrivia

http://www.cs.cornell.edu/courses/cs3410/2015sp

- 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
• cs-3410-staff-l@cornell.edu
• The email alias goes to me and the TAs, not to whole class

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

Newsgroup
• http://www.piazza.com/cornell/spring2015/cs3410
• For students

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

Lab Sections start **this** week

- This week: “hello world” lab: Intro to C and virtual machines
- Next week: Intro to logisim, logic circuits 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
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

Computer System = ?

Input +
Output +
Memory +
Datapath +
Control

Registers
CPU

Video

Network
USB

Keyboard
Mouse

Memory

Disk

Audio

Serial
```c
int x = 10;
x = 2 * x + 15;
```

**C compiler**
```
addi r5, r0, 10
muli r5, r5, 2
addi r5, r5, 15
```

**MIPS assembly language**
```
addi r5, r0, 10  \(\Rightarrow r5 = r0 + 10\)
muli r5, r5, 2  \(\Rightarrow r5 = r5 \times 2\)
addi r5, r5, 15  \(\Rightarrow r5 = r15 + 15\)
```

**MIPS machine language**
```
op = addi \hspace{0.5cm} r0 \hspace{0.5cm} r5 \hspace{0.5cm} 10
001000000000010100000000000001010
0000000000000010100101000001000000
001000001010010100000000000001111
```

```
op = addi \hspace{0.5cm} r5 \hspace{0.5cm} r5 \hspace{0.5cm} 15
001000001010010100000000000001111
```
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

```
01010000
10010100
...
```

```
memory
```
How to Design a Simple Processor

- Memory
- Instruction (inst) pointer
- Register file
- PC calculation
- Control
- ALU

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
  – coprocessor
• Floating Point
• 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>immediate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OP</td>
<td>jump target</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Overview

Application

Operating System

Compiler

Instruction Set Archaeology

Firmware

Memory system

Instr. Set Proc.

I/O system

Datapath & Control

Digital Design

Circuit Design
Applications

Everything these days!

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

- Xilinx FPGA
- NVidia GPU
- Cloud Computing
- Cell Phone
- Berkeley mote
- Cars

Bar chart showing the growth in millions of applications from 1997 to 2007:
- Cell Phones: 1193, 295, 405, 502, 785, 2029, 3650
- PCs: 114, 135, 136, 265, 650
- TVs: 93, 114, 135, 136, 202, 265, 650

Year:
- 1997
- 1999
- 2001
- 2003
- 2005
- 2007
Covered in this course

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

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