CS 3410: Computer System Organization and Programming

Hakim Weatherspoon

CS 3410, Spring 2013

Computer Science

Cornell University
Basic Building Blocks: A switch

A switch is a simple device that can act as a conductor or isolator

Can be used for amazing things...
NMOS and PMOS Transistors

- **NMOS Transistor**
  - Connect source to drain when $V_G = V_{\text{supply}}$
  - $V_D = 0V$
  - $V_G = V_{\text{supply}}$
  - $V_S = 0V$
  - Closed switch when $V_G = V_{\text{supply}}$

- **PMOS Transistor**
  - Connect source to drain when $V_G = 0V$
  - $V_D = V_{\text{supply}}$
  - $V_G = V_{\text{supply}}$
  - $V_S = V_{\text{supply}}$
  - Closed switch when $V_G = 0V$

- **N-channel transistor**
- **P-channel transistor**

---

$V_S$: voltage at the source  
$V_D$: voltage at the drain  
$V_{\text{supply}}$: max voltage (aka a logical 1)  
(ground): min voltage (aka a logical 0)
NMOS and PMOS Transistors

- **NMOS Transistor**
  - Connect source to drain when gate = 1
  - N-channel transistor

- **PMOS Transistor**
  - Connect source to drain when gate = 0
  - P-channel transistor

**Symbols and Definitions**
- $V_S$: voltage at the source
- $V_D$: voltage at the drain
- $V_{\text{supply}}$: max voltage (aka a logical 1)
- (ground): min voltage (aka a logical 0)
Inverter

- **Function**: NOT
- **Called an inverter**
- **Symbol**: 

V\text{supply} (aka logic 1)

<table>
<thead>
<tr>
<th>In</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Truth table

- CMOS: complementary-symmetry metal–oxide–semiconductor

(ground is logic 0)
NAND Gate

- Function: NAND
- Symbol:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
NOR Gate

- Function: NOR
- Symbol:
Building Functions

NOT:

AND:

OR:

NAND and NOR are universal
  • Can implement any function with NAND or just NOR gates
  • useful for manufacturing
Then and Now

The first transistor

• on a workbench at AT&T Bell Labs in 1947
• Bardeen, Brattain, and Shockley

An Intel Westmere

• 1.17 billion transistors
• 240 square millimeters
• 32 nanometer: transistor gate width
• Six processing cores
• Release date: January 2010

http://www.theregister.co.uk/2010/02/03/intel_westmere_ep_preview/
Then and Now

The first transistor
  • on a workbench at AT&T Bell Labs in 1947
  • Bardeen, Brattain, and Shockley

An Intel Ivy Bridge
  • 1.4 billion transistors
  • 160 square millimeters
  • 22 nanometer: transistor gate width
  • Up to eight processing cores
  • Release date: April 2012

The first transistor
• on a workbench at AT&T Bell Labs in 1947
• Bardeen, Brattain, and Shockley

Samsung Galaxy Note II
• Eynos 4412 System on a Chip (SoC)
• ARM Cortex-A9 processing core
• 32 nanometer: transistor gate width
• Four processing cores
• Release date: November 2012
Moore's Law

The number of transistors integrated on a single die will double every 24 months...
– Gordon Moore, Intel co-founder, 1965

Amazingly Visionary

1971 – 2300 transistors – 1MHz – 4004
1990 – 1M transistors – 50MHz – i486
2001 – 42M transistors – 2GHz – Xeon
2004 – 55M transistors – 3GHz – P4
2007 – 290M transistors – 3GHz – Core 2 Duo
2009 – 731M transistors – 2GHz – Nehalem
2012 – 1400M transistors – 2-3GHz – Ivy Bridge
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
Announcements: How class organized

Instructor: Hakim Weatherspoon (hweather@cs.cornell.edu)

Lecture:
• Tu/Th 1:25-2:40
• Olin 155

Lab Sections:
• Carpenter 104 (Blue Room)
• Carpenter 235 (Red Room)

Required Textbooks

Suggested Textbook
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
- Cornell National λ-Rail Rings testbed
- Software Defined Network Adapter
- Energy: KyotoFS/SMFS

Antiquity: built a global-scale storage system
Course Staff

cs3410-staff-l@cs.cornell.edu

Lecture/Homwork TA’s
- Detian Shi (ds629@cornell.edu)
- Paul Upchurch (paulu@cs.cornell.edu) (lead)
- Paul Heran Yang (hy279@cornell.edu)

Lab TAs
- Efe Gencer (gencer@cs.cornell.edu)
- Erluo Li (el378@cornell.edu)
- Han Wang (hwang@cs.cornell.edu) (lead)

Lab Undergraduate consultants
- Roman Averbukh (raa89@cornell.edu)
- Favian Contreras (fnc4@cornell.edu)
- Jisun Jung (jj329@cornell.edu)
- Emma Kilfoyle (efk23@cornell.edu)
- Joseph Mongeluzzi (jam634@cornell.edu)
- Sweet Song (ss2249@cornell.edu)
- Peter Tseng (pht24@cornell.edu)
- Victoria Wu (vw52@cornell.edu)
- Jason Zhao (jlz27@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 Phillips (PHL) 203
- Class started yesterday and ends March 1st

CS 2022 (Introduction to C)
- 1-credit course will greatly help with CS 3410
- Unfortunately, offered in the fall, not spring
- Instead, we will offer a primer to C next Monday, January 28th, 6-8pm. Location TBD.
# Grading

## Lab  
(45-50%)
- 5-6 Individual Labs  
  - 2 out-of-class labs  (15-17.5%)
  - 3-4 in-class labs  (5-7.5%)
- 4 Group Projects  (30%)
- Quizzes in lab  (2.5%)

## Lecture  
(45-50%)
- 3 Prelims  (32.5 - 37.5%)
  - Tue Feb 26th, Thur Mar 28th, and Thur Apr 25th
- Homework  (10%)
- Quizzes in lecture  (2.5%)

## Participation/Discretionary  (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

Published by AAAS
Adminstrivia

http://www.cs.cornell.edu/courses/cs3410/2013sp

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

Lab Sections (start today)

- Labs are separate than lecture and homework

- Bring laptop to Labs (optional)
Administrivia

http://www.cs.cornell.edu/courses/cs3410/2013sp

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

Lab Sections (start today)

<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>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>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: intro to logisim and building an adder
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
• http://www.piazza.com/cornell/spring2012/cs3410
• For students

iClicker
• http://atcsupport.cit.cornell.edu/pollsrvc/
Lab Sections start *this* 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
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 me 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

- Registers
- CPU

- 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
muli r5, r5, 2
addi r5, r5, 15

MIPS machine language

001000000000010101000000000000001010
0000000000000101001010010000100000
00100001010010100000000000001111
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

```
| regs | processor | bus | | | addr, data, r/w | memory |
|------|-----------|-----|-----|---------|--------|
|      |           |     | 01010000 | 10010100 | ...    |
```
How to Design a Simple Processor

memory

inst

32

pc

00

new pc calculation

register file

control

alu

00: addi r5, r0, 10
04: multi 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
- 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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OP</th>
<th>rs</th>
<th>rt</th>
<th>immediate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OP</th>
<th>jump target</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Applications

Everything these days!
  • Phones, cars, televisions, games, computers,...
Covered in this course

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