Instructors

- Professor: Daniel J. Sorin
  - Office: Hudson Hall 1111 (that’s 1510, if you prefer binary)
  - Email: sorin@ee.duke.edu
  → subject of all emails to me must begin with ECE152
  - Office Hours: Tues 3-4, Fri 1-2
    - If you have class at both of these times, email me to meet

- Graduate Teaching Assistant: Nick Kirchem
  - Email: nrk2@ee.duke.edu
  - Office Hours: Weds 11-12, Fri 1-2
  - "Office": Hudson 01B (ECE 151/152 lab)
  - Roles: office hours, answering questions, etc.

Undergrad Teaching Assistants

- Undergraduate TAs (UTAs):
  - Chris Abbott (cba4@duke.edu)
  - Thurs 6-8pm @ Hudson 01B
  - Raj Baba (rb3@ee.duke.edu)
  - No office hours
  - Andrew Dreher (ad013@ee.duke.edu)
  - Thurs 1:10-2:30pm @ CIEMAS 2nd floor foyer
  - Trina Kok (tk16@duke.edu)
  - Mondays 2-4pm @ CIEMAS 2nd floor foyer

- Roles
  - Answering email questions about homeworks and project
  - Holding office hours to help with CAD software

Course Website

- Course Web Page
  - http://www.ee.duke.edu/~sorin/ece152/
  - Lecture slides available on web before or shortly after class
  - Print them out and bring them with you to class
  - I intentionally leave out important stuff that we cover in class!
  - Missing class = missing important information
  - All important info for course is on website
  - Please check it before emailing me or TA
  - You are required to monitor web page
  - Homework and project assignments will appear on web page
  - I will announce them as well

Textbook

- Text: Computer Organization & Design (Patterson & Hennessy)
  - 3rd edition of the textbook
  - You are expected to complete the assigned readings

- We will not cover chapters in the textbook in a strictly linear fashion
  - We will finish all of the fundamental material before digging into supplemental materials
Workload

- Readings from textbook
- Homework assignments (performed individually)
  - Pencil and paper problems
  - Programming problems
- Project (performed in groups of 3)
  - Software simulator of the MIPS 152/16 computer
  - Building the MIPS 152/16 computer in real hardware!
  - I have assigned the groups and I will ensure that group grading is done fairly
  - Group assignments already posted on course newsgroup
  - Project will be broken up into smaller parts to make it more manageable
  - Project Part 1: Building a register file, due Monday Jan. 24th

Grading

- Grade breakdown
  - Homework 30%
  - Project 25% (graded as a group, but fairly)
  - Midterm Exam 20%
  - Final Exam 25%
- I strongly believe in partial credit
  - Please explain your answers to get as much credit as possible
- Late homework policy
  - 10% reduction for each day late
  - No credit after the homework is graded and handed back
- Assignments take a lot of time, so start them early
  - Yes, this means you!

Academic Misconduct

- Academic Misconduct
  - Refer to Duke Honor Code (link to it off course website)
  - Studying together in groups is encouraged
  - But all homework must be your own, unless otherwise stated
  - Common examples of cheating:
    - Running out of time and then picking up someone else's output
    - Borrowing code from someone who took course before
    - Using solutions found on the Web
    - Person asks to borrow solution “just to take a look”
    - Copying an exam question
- I will not tolerate any academic misconduct!
  - Historically, this course has "led the league" in cases of academic misconduct that have led to suspensions and expulsions
  - Software for detecting cheating is very, very good … and I use it

Goals of This Course

- By end of semester, you will
  - Know how computers work
  - What's inside a computer?
  - How do computers run programs written in C/C++/Java?
  - Design your own computers, simulate their behavior, and build them in real hardware!
  - Understand the engineering tradeoffs to be made in the design and implementation of different types of computers

Outline of Introduction

- Administrivia
- What is computer architecture?
  - What’s inside a computer?
  - Kinds of computers
  - Trends in computer architecture
- What do computers do?
- Representing high level things in binary

Reading Assignment

- Patterson & Hennessy
  - Chapter 1
    - This is a short and relatively easy-to-read chapter
    - Please read it such that afterwards you’d feel comfortable teaching the material to an ECE 151 student
  - For those of you who haven’t done digital logic design in a while, you may want to go back to your ECE 151 textbook or skim Appendix B of Patterson & Hennessy
    - Digital logic design is a prerequisite for this course, but I understand if some of you haven’t done this in a while
What is a Computer?
- What kinds of computers are there?
- What do computers do?
- What's inside a computer?

Kinds of Computers
- "Traditional" personal computers
  - Laptop, desktop
- Less-traditional personal computers
  - PDA, iPod, Gameboy, cell phone
- Hidden big computers
  - Mainframes and servers for business, science, government
- Hidden embedded computers
  - Controllers for cars, airplanes, toasters, VCRs, etc.
  - Far and away the largest market for computers!

The Inside of a Computer
- The Five Classic Components of a Computer

System Organization

What is Computer Architecture?
- Architecture = interface between hardware and software
- ECE 152 = design of CPU, memory, and I/O

Forces on Computer Architecture
- Technology
- Programming Languages
- Applications
- Operating Systems
- History
A Very Brief History of Computing

- 1645 Blaise Pascal’s Calculating Machine
- 1822 Charles Babbage
  - Difference Engine
  - Analytic Engine: Augusta Ada King first programmer (woman)
- < 1946 Eckert & Mauchly
  - ENIAC (Electronic Numerical Integrator and Calculator)
- 1947 John von Neumann
  - Proposed the Stored Program Computer
  - Properties of today’s computers
- 1949 Maurice Wilkes
  - EDSAC (Electronic Delay Storage Automatic Calculator)

Commercial Computers

<table>
<thead>
<tr>
<th>Year</th>
<th>Name</th>
<th>Size (cu. ft.)</th>
<th>Adds/sec</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1951</td>
<td>UNIVAC I</td>
<td>1000</td>
<td>1,900</td>
<td>$1,000,000</td>
</tr>
<tr>
<td>1964</td>
<td>IBM 704</td>
<td>60</td>
<td>500,000</td>
<td>$1,000,000</td>
</tr>
<tr>
<td>1965</td>
<td>PDP-8</td>
<td>8</td>
<td>330,000</td>
<td>$16,000</td>
</tr>
<tr>
<td>1976</td>
<td>Cray-1</td>
<td>58</td>
<td>166,000</td>
<td>$4,000,000</td>
</tr>
<tr>
<td>1981</td>
<td>IBM PC</td>
<td>240,000</td>
<td>2</td>
<td>$3,000</td>
</tr>
<tr>
<td>1991</td>
<td>HP 9000 / model 750</td>
<td>2</td>
<td>50,000,000</td>
<td>$7,400</td>
</tr>
<tr>
<td>1996</td>
<td>Intel PentiumPro</td>
<td>2</td>
<td>400,000,000</td>
<td>$4,400</td>
</tr>
<tr>
<td>2002</td>
<td>Intel Pentium4</td>
<td>0.25</td>
<td>4,000,000,000</td>
<td>$1,500</td>
</tr>
</tbody>
</table>

Microprocessor Trends

What Do Computer Architects Do?

- Architects design computers that meet ever-changing needs and challenges
  - Tailored to new applications (e.g., image/video processing)
  - Amenable to new technologies (e.g., faster and more plentiful transistors)
- Computer architecture is engineering, not science
  - There is no one right way to design a computer → this is why there isn’t just one type of computer in the world
  - This does not mean, though, that all computers are equally good
- Full disclosure: I’m a computer architect

Outline of Introduction

- Administrivia
- What is computer architecture?
- What do computers do?
- Representing high level things in binary

Computers Do What You Tell Them To Do!

- Computers aren’t very smart
  - They can only do what you tell them to do
- We use software to tell them what to do
  - Software is made up of lots of instructions
  - Computers run software → computers execute instructions
- In fact, computers can only execute instructions that are in their specific machine language
- Every type of computer (i.e., computer architecture) has a different set of instructions that it understands
  - Intel IA-32 (e.g., Pentium3, Pentium4)
  - Intel IA-64 (Itanium, Itanium 2)
  - PowerPC (in Apple Macs)
  - SPARC (in Sun computers)
  - Etc.
Computers Execute Instructions

- What kinds of instructions are there?
  - Arithmetic: add, subtract, multiply, divide, etc.
  - Read/write from memory
  - Conditional: if condition, then jump to other part of program
  - What other kinds of instructions might be useful?
- How do we represent instructions?
  - Digitally! With strings of zeros and ones
- A computer is just a digital system
  - Consists of combinational and sequential digital logic
  - Building a computer would be a very exciting ECE 151 project
  - So how do computers run programs in Java or C?
    - None of us write programs in binary (zeros and ones) ...

Levels of Representation

- High Level Language
  - Programs are written in English-like languages
  - Examples: Java, C, C++, Fortran, Basic, Pascal, Lisp, Ada, etc.
  - These languages are easy for humans to understand
- Assembly Language
  - Programs are written in low-level languages
  - These languages are difficult for humans to understand
- Machine Language
  - Programs are written in binary (zeros and ones)
  - These languages are not easy for humans to understand

- Every computer architecture has its own assembly language
- Assembly languages tend to be pretty low-level, yet some actual humans still write code in assembly
- But most code is written in high level languages and compiled
  - Compiler is a program that automatically converts HLL to assembly

Levels of Representation

- High Level Language
  - Programs are written in English-like languages
  - Examples: Java, C, C++, Fortran, Basic, Pascal, Lisp, Ada, etc.
  - These languages are easy for humans to understand
- Assembly Language
  - Programs are written in low-level languages
  - These languages are difficult for humans to understand
- Machine Language
  - Programs are written in binary (zeros and ones)
  - These languages are not easy for humans to understand

- There are many high level languages (HLLs)
  - Java, C, C++, Fortran, BASIC, Pascal, Lisp, Ada, etc.
  - HLLs tend to be English-like languages that are "easy" for programmers to understand
  - In this class, we'll focus on C/C++ as our running example for HLL code.
    - Why? C/C++ has pointers
    - C/C++ has explicit memory allocation/deallocation

Outline of Introduction

- Administrivia
- What is computer architecture?
  - What do computers do?
  - Representing high level things in binary
    - Data objects: integers, decimals, characters, etc.
  - Memory locations
  - (We'll get to instruction representations a bit later in course)