ECE 250 / CPS 250
Computer Architecture

Introduction

Benjamin Lee
Slides based on those from
Andrew Hilton (Duke), Alvy Lebeck (Duke)
Benjamin Lee (Duke), and Amir Roth (Penn)
Instructor and Graduate TAs

• Professor: Benjamin Lee
  • Office: Hudson Hall 210
  • Email: benjamin.c.lee@duke.edu
  • Office Hours: TuTh 4-5PM or by appointment

• Graduate TAs:
  • Alfredo Velasco (alfredo.velasco@duke.edu)
  • Pengfei Zheng (pengfei.zheng@duke.edu)
Undergrad Teaching Assistants

• Undergraduate TAs (UTAs)
  • A whole bunch of awesome undergrads who aced this class

• Will help with
  • Leading recitations
  • Answering questions about homeworks
  • Holding office hours to help with tools and software

• Will NOT bail you out at 3am when deadline is at 10am
Getting Info

• Course Web Page
  http://people.duke.edu/~bcl15/class/class_ece250fall15.html
  • syllabus, schedule, rules/policies, Prof/TA info, office hour info
  • lecture slides, links to useful resources

• Sakai: dynamic info
  • From me: announcements, assignments, grades
  • From you: uploaded homeworks

• Piazza: questions/answers
  • Post all of your questions here
  • Questions must be “public” unless very good reason otherwise

Consult these resources in this order
Notes About Lectures and Lecture Slides

• Lecture slides available on Sakai before class
  • Print them out and bring them with you to class
  • Value (just reading slides) << Value (attending class)
  • Missing class = missing important course material

• Lectures will be recorded on Panopto
  • Useful if you’re sick or out of town or if you want to review a previously attended lecture at your own pace
  • Value (watching on Panopto) < Value (attending class)
Getting Answers to Questions

• There are too many students for you all to email me
  • So now what do you do if you have a question?
1. Check the course website
2. Check Sakai
3. Check Piazza
  • If you have questions about homeworks, use Piazza – then everyone can see the answer(s) posted there by me, a TA, or your fellow classmate
  • Professor and TAs will NOT answer direct emails about homeworks or anything that pertains to more than 1 student
• Contact TA directly if: grading issue
• Contact professor directly if issue that is specific to you and that can’t be posted on Piazza (e.g., missing exam)
Textbook

- **Text**: *Computer Organization & Design* (Patterson & Hennessy)
  - 5th edition of the textbook
  - You are expected to complete the assigned readings

- We will not cover material in the textbook in a strictly linear fashion
Other Resources

• There are many online resources, including:
  • Unix tutorials
  • C programming tutorials
  • Videos of Prof. Hilton (Duke ECE/CS) teaching C programming
  • Coursera course on computer architecture
  • Etc.

• Many useful links on course website
• Feel free to use these materials, but none are required
Workload

- Readings from textbook
- Homework assignments – **done individually**
  - Pencil and paper problems
  - Programming problems in C and assembly
  - Digital logic design problems (like designing a computer)
- Recitations – **done with partners**
  - During recitations, work with partners/groups (or individually) on “assignments” to help you learn skills useful for homeworks and tests
  - Goal: learning through hands-on, low-stress practice
  - UTAs will help students during recitations
  - Bring a laptop to work on – or work with a partner who has one
Grading

• Grade breakdown
  • Homework 50%
  • Midterm #1 12.5%
  • Midterm #2 12.5%
  • Final Exam 25%

• I strongly believe in partial credit
  • Please explain your answers to get as much credit as possible

• Late homework policy – no exceptions, no extensions
  • 0-24 hours late: 10% penalty
  • 24-48 hours late: 20% penalty
  • >48 hours late: no credit

• Assignments take a lot of time, so start them early
Academic Misconduct

- Duke Community Standard
  - Homework is individual – you must do your own work
  - You violate academic integrity when
    - you obtain solutions and code from others, or
    - you provide solutions and code to others
  - Common examples of cheating:
    - Running out of time and using someone else's output
    - Borrowing code from someone who took the course before
    - Using solutions found on the Web
    - Having a friend help you to debug your program

- I will not tolerate any academic misconduct!
  - Software for detecting cheating is very, very good ... and I use it
  - 12 students were caught on Homework #1 in spring 2014
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, Matlab, etc.?
  • You will design hardware that computers use
  • You will understand the engineering tradeoffs to be made in the design of different types of computers
  • You will know how to program in C
  • You may, like me, decide to become an architect. 😊

• If, at any point, it’s not clear why I’m talking about some topic, please ask!
Outline of Introduction

• Administrivia
• What is a computer?
• What is computer architecture?
• Why are there different types of computers?
• What does the rest of this course look like?
Reading Assignment

- Patterson & Hennessy
  - Chapter 1
  - This is a short and relatively easy-to-read chapter
What is a Computer?

- A computer is just a machine
  - A bunch of switches and logic that we’ll talk about later
- Yes, but what does this machine do?
  - Whatever you tell it to do! No more, no less
- A computer just does what software tells it to do
  - Software is a series of instructions
- ICQ (In-Class Question): What instructions does a computer need?
Computers Execute Instructions

• What kinds of instructions are there?
  • Arithmetic: add, subtract, multiply, divide, etc.
  • Access memory: read, write
  • Conditional: if condition, then jump to other part of program
  • What other kinds of instructions might be useful?

• So how do computers run programs in Java or C/C++ or Matlab or ....?
  • None of us write programs in binary (zeros and ones) ...
  • We’ll get to this in a few minutes
Instruction Sets

• Computers can only execute instructions that are in their specific machine language

• Every type of computer has a different instruction set that it understands
  • Intel (and AMD) IA-32 (x86): Pentium Core i7, AMD Opteron, etc.
  • ARM: In many embedded processors (e.g., smartphones)
    • ISA used by many companies (e.g., Qualcomm)
  • Intel IA-64: Itanium, Itanium 2
  • PowerPC: In Cell Processor and old Apple Macs
  • SPARC: In computers from Sun Microsystems/Oracle
  • MIPS: MIPS R10000 → this is the example used in the textbook

• Note: no computer executes Java or C++
Outline of Introduction

- Administrivia
- What is a computer?
- What is computer architecture?
- Why are there different types of computers?
- What does the rest of this course look like?
Hint: It Doesn’t Involve Skyscrapers …

• Strictly speaking, a **computer architecture** specifies what the hardware looks like (its interface), so that we can write software to run on it
  - Exactly what instructions does it have
  - Number of storage locations it has
  - And more that we’ll learn about later in semester

• **Important point:** there are many, many different ways to build machines that provide the same interface to software
  - There are many **microarchitectures** that conform to same architecture
  - Some are better than others! If you don’t believe me, I’ll trade you my original Intel Pentium for your Intel Core i7

• **ICQ:** So what’s inside one of these machines?
The Inside of a Computer

• The Five Classic Components of a Computer

Diagram showing the five classic components of a computer: Processor/CPU, Control, Datapath, Memory, Input, Output.
System Organization

Diagram showing the system organization, including components such as CPU, cache, memory bus, main memory, I/O bridge, I/O Bus, disk controller, graphics controller, network interface, disk, and graphics. The diagram also includes a network connection.
What Is ECE/CS 250 All About?

• Architecture = interface between hardware and software

![Diagram showing the layers of software and hardware]

• ECE/CS 250 = design of CPU, memory, and I/O
• ECE/CS 350 = building it in hardware
Outline of Introduction

• Administrivia
• What is a computer?
• What is computer architecture?
• Why are there different types of computers?
• What does the rest of this course look like?
Differences Between Computers

• We have different computers for different purposes

• Some can achieve performance needed for high-performance gaming
  • E.g., Cell Processor in PlayStation 3

• Others can achieve decent enough performance for laptop without using too much power
  • E.g., Intel Atom for Mobile

• And yet others can function reliably enough to be trusted with the control of your car’s brakes

ICQ: What computers do you use?
ICQ: Which of those computers do you own?
Kinds of Computers

• “Traditional” personal computers
  • Laptop, desktop, netbook

• Less-traditional personal computers
  • iPad, iPhone, Samsung/Android smartphone, iPod, Xbox, etc.

• Hidden “big” computers (some are in the “cloud”)
  • Mainframes and servers for business, science, government
    • E.g., the machines that run Duke email, ACES, etc.
  • Google has many thousands of computers (that you don’t see)

• Hidden embedded computers
  • Controllers for cars, airplanes, ATMs, toasters, DVD players, etc.
    • Far and away the largest market for computers!

• Other kinds of computers??
Forces on Computer Architecture

- Technology
- Programming Languages
- Applications
- Operating Systems
- History

Computer Architecture

© Daniel J. Sorin
from Hilton, Lebeck, Lee, Roth
A Very Brief History of Computing

- 1645 Blaise Pascal’s Calculating Machine
- 1822 Charles Babbage
  - Difference Engine
  - Analytic Engine: Augusta Ada King first programmer
- < 1946 Eckert & Mauchly
  - **ENIAC** (Electronic Numerical Integrator and Calculator)
- 1947 John von Neumann
  - Proposed the Stored Program Computer
  - Virtually all current computers are “von Neumann” machines
- 1949 Maurice Wilkes
  - **EDSAC** (Electronic Delay Storage Automatic Calculator)
## Some 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 S/360 Model 50</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>1965</td>
<td>Cray-1</td>
<td>58</td>
<td>166 million</td>
<td>$4,000,000</td>
</tr>
<tr>
<td>1981</td>
<td>IBM PC</td>
<td>desktop</td>
<td>240,000</td>
<td>$3,000</td>
</tr>
<tr>
<td>1991</td>
<td>HP 9000 / model 750</td>
<td>desktop</td>
<td>50 million</td>
<td>$7,400</td>
</tr>
<tr>
<td>1996</td>
<td>PC with Intel PentiumPro</td>
<td>desktop</td>
<td>400 million</td>
<td>$4,400</td>
</tr>
<tr>
<td>2002</td>
<td>PC with Intel Pentium4</td>
<td>desktop/laptop/rack</td>
<td>4 billion</td>
<td>$1-2K</td>
</tr>
<tr>
<td>2008</td>
<td>Cell processor</td>
<td>PlayStation3</td>
<td>~200 billion</td>
<td>~$350 (eBay)</td>
</tr>
<tr>
<td>2014</td>
<td>Nvidia K40 GPU</td>
<td>Desktop/rack</td>
<td>~4.3 trillion</td>
<td>$4,000</td>
</tr>
</tbody>
</table>
Microprocessor Trends (for Intel CPUs)

from Hilton, Lebeck, Lee, Roth

ECE/CS 250
What Do Computer Architects Do?

• Full disclosure: I’m a computer architect
• Design new microarchitectures
  • Very occasionally, we design new architectures
• 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)
  • More reliable, more secure, use less power, etc.
• 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
What You Will Learn In This Course

• The basic operation of a computer
  • Primitive operations (instructions)
  • Computer arithmetic
  • Instruction sequencing and processing
  • Memory
  • Input/output
  • Doing all of the above, just faster!

• Understand the relationship between abstractions
  • Interface design
  • High-level program to control signals (SW \rightarrow HW)

• C programming \rightarrow why?
Course Outline

- Introduction to Computer Architecture
- C Programming and From C to Binary (next!)
- Instruction Sets & Assembly Programming
- Processor Core Design
- Memory Systems
- I/O Devices and Networks
- Pipelined Processor Cores
- Multicore Processors
The Even Bigger Picture

- ECE/CS 250: Basic computer design
  - Finish 1 instruction every 1 very-long clock cycle
  - Finish 1 instruction every 1 short cycle (using pipelining)
- ECE/CS 350: Implementing digital computers/systems
- ECE 552/CS 550: High-performance computers + more
  - Finish ~3-6 instructions every very-short cycle
  - Multiple cores each finish ~3-6 instructions every very-short cycle
  - Out-of-order instruction execution, power-efficiency, reliability, security, etc.
- ECE 652/CS 650: Highly parallel computers and other advanced topics
- ECE 554/CS ????: Fault tolerant computers