Introduction

Tyler Bletsch
Duke University

Slides are derived from work by
Daniel J. Sorin (Duke), Andrew Hilton (Duke), Alvy Lebeck (Duke),
Benjamin Lee (Duke), and Amir Roth (Penn)
Course objective: Evolve your understanding of computers

Before

Input

Good stuff

Also bad stuff

And weird stuff

Also it hangs some times I dunno wtf is up with that
Course objective: Evolve your understanding of computers

After

Input

Output

No more magic!
Instructor and Graduate TAs

• Professor: Tyler Bletsch
  • Office: Hudson Hall 106
  • Email: Tyler.Bletsch@duke.edu
  • Office Hours: TBD

• Graduate TAs:
  • Alfredo Velasco (alfredo.velasco@duke.edu)
  • Xavier De Gunten (xavier.de.gunten@duke.edu)
  • Bryan Prosser (bryan.prosser@duke.edu)

• Undergraduate TAs:
  • Everyone ever
Undergrad Teaching Assistants

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

- Will help with
  - Answering email 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: static info
  - [http://people.duke.edu/~tkb13/courses/ece250/](http://people.duke.edu/~tkb13/courses/ece250/)
    - Syllabus, schedule, slides, assignments, rules/policies, prof/TA info, office hour info
    - Links to useful resources

- Piazza: questions/answers
  - Post all of your questions here
  - Questions must be “public” unless good reason otherwise
    - **No code** in public posts!

- Sakai: just assignment submission and gradebook
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 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

  - Not the “ARM edition” or “Revised Printing” or whatever

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

If you go to addall.com, you can search all online booksellers at once. Amazon price for text: $66.50 used. Addall found it for $56.
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

• 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 on exercises to help you learn skills necessary for homeworks and exams. Can also get homework help once done
  • UTAs will help students during recitations
  • **Bring a laptop to work on** – if you don’t have one, please with a partner who has one or contact me about getting a loaner
Lecture vs. Recitation

- **Lecture:**
  - Learning the theory the underlies computers
  - **Necessary to achieve understanding and do well in the course**
  - Attendance expected but not tracked

- **Recitation:**
  - Learning practical skills needed to understand and design computers
  - **Necessary to achieve understanding and do well in the course**
  - Attendance required (with one exception). Grading:
    - Students attending and making *good faith effort* will receive full credit for the day.
    - You may miss a week's recitation if the exercise for that week is completed and submitted by 8am on the morning of the recitation. *Good faith effort* is required.
Lecture vs. Recitation

- If you attend lecture but not recitation:
  - You won’t know how to *do* the assignment
Lecture vs. Recitation

- If you attend recitation but not lecture:
  - You won’t know *how* to do the assignment
Lecture vs. Recitation

- If you attend recitation AND lecture:
  - Your hands will turn into creepy robot hands but you’ll probably get a good grade
# Grading Breakdown

<table>
<thead>
<tr>
<th>Assignment</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Homeworks</td>
<td>50%</td>
</tr>
<tr>
<td>Recitation</td>
<td>5%</td>
</tr>
<tr>
<td>Exam 1</td>
<td>10%</td>
</tr>
<tr>
<td>Exam 2</td>
<td>10%</td>
</tr>
<tr>
<td>Final Exam</td>
<td>25%</td>
</tr>
</tbody>
</table>

Partial credit is available – provide detail in your answers to seek it!

Late homework submissions incur penalties as follows:
- Submission is 0-24 hours late: total score is multiplied by 0.9
- Submission is 24-48 hours late: total score is multiplied by 0.8
- Submission is more than 48 hours late: total score is multiplied by the Planck constant \((\text{in J} \cdot \text{s})\)

NOTE: If you feel *in advance* that you may need an extension, contact the instructor.

These assignments are looooolllllllllllloooong. START EARLY.
Grade Appeals

• All regrade requests must be in writing
  • Email the UTA who graded the question (we’ll indicate who graded what)

• After speaking with the TA, if you still have concerns, contact the instructor

• All regrade requests must be submitted no later than 2 weeks after the assignment was returned to you.
Academic Misconduct

• Academic Misconduct
  • Refer to Duke Community Standard
  • Homework is individual – you do your own work
  • Common examples of cheating:
    • Running out of time and using someone else's output
    • Borrowing code from someone who took 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
  • 8 students were busted on Homework #1 in spring 2013, and 2 of them were referred to the Office of Student Conduct

• “But I didn’t know that was cheating” is not a valid excuse
MOSS: Measure of Software Similarity

Doesn't care about:
- Comments
- Whitespace
- Naming
- Values

Only cares about code structure.

How to beat it?
Write your own code
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

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

• The instructor and TA will...
  • Provide lectures/recitations at the stated times
  • Set clear policies on grading
  • Provide timely feedback on assignments
  • Be available out of class to provide reasonable assistance
  • Respond to comments or complaints about the instruction provided

• Students are expected to...
  • Receive lectures/recitations at the stated times
  • Turn in assignments on time
  • Seek out of class assistance in a timely manner if needed
  • Provide frank comments about the instruction or grading as soon as possible if there are issues
  • Assist each other within the bounds of academic integrity
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?
What is a Computer?

- A machine that follows simple instructions deterministically.

- It just does what software tells it to do
  - Software is a series of these instructions

- 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 whatever the cool kids are using these days?
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 (incl. Sony PS3) 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?
Computer Architecture

- **Computer architecture** specifies what the hardware looks like (its interface), so that we can write software to run on it
  - What instructions does it have?
  - Number of storage locations it has?
  - More stuff (covered later...)

- **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

- What’s inside one of these machines?
All computers are like fast food restaurants

• Fast Food Architecture: the interface
  • Menu
  • How/where to place orders
  • How finished orders are given to customers

• Fast Food Microarchitecture: the implementation
  • What ingredients are used
  • What appliances are available
  • How many employees you have and what they do
The Inside of a Computer

- The Five Classic Components of a Computer

- Processor/CPU
- Control
- Datapath
- Memory
- Input
- Output
What Is ECE/CS 250 All About?

• Architecture = interface between hardware and software

• 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 for high-performance gaming
  • E.g., Cell Processor in PlayStation 3

• Some for power-efficiency at acceptable performance (laptop)
  • E.g., Intel Pentium M (for Mobile)

• Some for extreme reliability
  • E.g., the CPU that runs your car’s brakes

What computers do you use?

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

Computer Architecture

- Technology
- Programming Languages
- History
- Applications
- Operating Systems
A Very Brief History of Computing

- 1645 Blaise Pascal’s Calculating Machine
- 1822 Charles Babbage
  - Difference Engine
  - Analytic Engine: Augusta Ada King (Lovelace), 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>1976</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)
What Do Computer Architects Do?

- 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 → HW)

• C programming → 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
• 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