Introduction

Tyler Bletsch
Duke University

Slides are derived from work by Andrew Hilton (Duke)
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!
Overview

• For: MS/MEng students who want Comp Eng focus..
  • ...but don’t have Comp Eng undergrad

• Background for
  • **ECE 650: Systems Programming and Engineering**
    • ECE 558: Computer Networks/Distributed Systems
    • CS 510: Operating Systems
    • ...
  • ECE 522: Advanced Computer Architecture
    • Co-req for Performance, Optimization, and Parallelism
    • Pre-req for 652
  • ECE 554: Fault-Tolerant and Testable Computer Systems
  • ECE 559: Advanced Digital System Design
What we will learn: 10K feet

- Transistors -> Processor
  - Logic gates, combinational logic, sequential logic, FSMs
  - Adders, multipliers, shifters
  - Latches, Flip-flops, SRAMs, DRAMs, CAMs
  - Single-cycle datapaths, pipelining
  - Caches, memory hierarchy, virtual memory
  - Interrupts, exceptions, IO

- Hardware/software interface (ISA)
  - MIPS assembly

- Operating System basics
  - System calls, protection, multi-tasking, ...

- Networking basics
  - 7 layer OSI model, TCP/IP, routing,...
How We Will Learn It

- Must “learn by doing”:
  - 4/5 homeworks: Implement something VHDL
- Write VHDL, synthesize it
  - Load it on *Altera DE2 board*
  - Run it, demo it to TAs
- 1/5 homework: Write MIPS assembly
  - Run it in SPIM
A Word About Varying Backgrounds

- I expect wide variations in backgrounds for this class
  - E.g., some know VHDL, some have never seen it
- Even if you are familiar with a topic we are covering...
  - You may learn something new
  - You may refresh rusty memory
  - I may present it slightly differently than you are used to
  - You may be able to help other students learn it
- If you missing some background (feeling lost)
  - Please come talk to me or a TA sooner rather than later!
Instructor and TAs

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

- **TAs:**
  - Peter Murphy (recitation lead) - peter.murphy2@duke.edu
  - Xiaobin Lin - xiaobin.lin@duke.edu
  - Qi Xu - xu.qi@duke.edu
  - Yiqiao Liu - yiqiao.liu@duke.edu

Note: TAs are NOT under obligation to bail you out at 3am. Your best bet is to get help in a timely manner!
Getting Info

- Course Web Page: static info
    - 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

- 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
- 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.
Homeworks

- Homeworks: 5 of them in the semester
  - Work in groups of 2 or 3
    - Fixed for semester once formed
    - Exceptional circumstances/dysfunctional group: talk to me
  - Do not work alone:
    - First assignment may be easy... but will want a group later
  - End of recitation this week: a few minutes to form groups

- “Demo” portion of homework
  - “Question and Answer” may be better term
  - TAs will ask each person questions about project: accountability
  - ALL group members MUST know how it all works
    - TAs will ask a particular group member a question
      - “I don’t know, John did that part” will lose points

TAs are specifically instructed to identify weaknesses in group member understanding! Thwart them by studying every aspect of the work!
Homework submission

• Each assignment is submitted to Sakai by one group member.
  • Multiple submissions by different students in a group? We’ll grade an arbitrary one 😞

• Late Policy
  • 5 days per group total for the semester
    • Does not change demo deadline, only submission deadline
  • Days, not classes
  • Used in entire days: 10 min late = on next day
  • After used up: must turn in on time
    • No credit for late work after this

• Extenuating circumstances: talk to me
  • E.g., serious injury/illness, family emergency...

Homework assignments are loooooooooooong. START EARLY.
Recitations

• Recitations – same groups as homeworks
  • During recitations, work on exercises to help you learn skills necessary for homeworks and exams. Can also get homework help once done
  • TA 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 that underlies computers
  - **Necessary to achieve understanding and do well in the course**
  - Attendance required, tracked intermittently by quizzes.

- **Recitation:**
  - Learning practical skills needed to understand and design computers
  - **Necessary to achieve understanding and do well in the course**
  - Attendance required. Grading:
    - Students attending and making *good faith effort* will receive full credit for the day.
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>Class/recitation attendance</td>
<td>10%</td>
</tr>
<tr>
<td>Homework</td>
<td>30%</td>
</tr>
<tr>
<td>Midterm</td>
<td>25%</td>
</tr>
<tr>
<td>Final exam</td>
<td>35%</td>
</tr>
</tbody>
</table>
Grade Appeals

• All regrade requests must be in writing
  • Email the TA 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 1 week after the assignment was returned to you.
Academic Misconduct

• Academic Misconduct
  • Refer to Duke Community Standard
  • Homework is group-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**
Academic Integrity: Homework

- Homework: You + your group
  - Should not be getting help from other groups, students
  - Can ask TAs + me for help
  - Not many external resources you should be using
Academic Integrity: Mini-Quizzes

- Mini-Quizzes: Individual Effort
  - Open Notes
    - Can use paper resources
    - Will allow electronic reading of course lecture slides
      - But may not use laptop/tablet for anything else
      - If you use an electronic device during the quiz, I’d better see the course lecture slides on it
  - May not discuss with classmates
  - Closed Book
Academic Integrity: Exams

- Exams in this class are individual effort
  - No notes/book/internet/peers

- Related exam policies:
  - Questions? Raise hand, TA or I will come to you (don’t get up)
  - Need restroom? Raise hand, we will let you go one at a time
  - No calculators/smart phones: too easy to use to chat
Academic Integrity: General

• Some general guidelines
  • If you don’t know if something is OK, please ask me.
  • If you think “I don’t want to ask, you will probably say no” that is a good sign its NOT acceptable.
  • If you do something wrong, and regret it, please come forward—I recognize the value and learning benefit of admitting your mistakes. (Note: this does NOT mean there will be no consequences if you come forward).
  • If you are aware of someone else’s misconduct, you should report it to me or another appropriate authority.
    • Within your homework group, this becomes even stronger: if you are aware that one of your group members has committed misconduct on a homework submission for your group, you are complicit in it if you do not report it.
Course Problems

- Can’t make midterms / final, other conflicts
  - Tell us early and we will schedule alternate time
- Struggling in course
  - Come see me: I’m here to help
- Other problems:
  - Feel free to talk to me, I’m generally understanding and will try to work with you
  - Some problems may extend well beyond my course
    - Academic Advisor
    - DGS Team
Our Responsibilities

• The instructor and TAs 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*
Wisdom From Prior Courses

• Some things I have learned from prior students
  • Cultural barriers to asking for help:
    • No question is “too simple” to ask me
    • Office hours are not just for the advanced students to ask complicated questions
    • You will not offend me if you ask me questions/tell me you do not understand
    • You are not “bothering me” if you come to office hours
      • That is why I have them: to help you!
    • Applies very broadly: I’m here to teach you and help you!
  • Doing badly -> “study harder”
    • Not the answer!
    • Work smarter, not harder.
    • Studying may not help at all!
    • Struggling? Seek my help.
Advice from past students of the undergrad version of this course

From a survey given at the end of Spring 2016.

• “Get started on the homework early.”
  • “start every assignment early! TA hours get so crowded the week before it's due and they are super helpful.”
  • “Start every homework as early as possible. It is almost impossible to judge how long they are going to take until you are a decent way through them. By starting early, you'll relieve some stress, and be able to approach teachers/TA's with tangible progress and questions early on. This makes the whole assignment go faster.”
  • “To go to office hours (usually early, they're really crowded) and to start on homeworks early too”
  • More...

• “Don't be intimidated if the information makes no sense at first. Be patient, use your resources, and take your time. It'll make sense. Going hand in hand with this, if you're going to take your time, start early on assignments to make sure you have that time.”

• “The textbook is actually really helpful, like definitely make an effort to read the pages that they assign. They explain things for the homework in more detail.”
Transistors => Processors

- Processors are made of **transistors**
  - Electrically controlled switches (more on this later)
  - Anyone have any idea how many transistors are in a modern chip?
Microprocessor Trends (for Intel CPUs)
Transistors => Processors

- Processors are made of **transistors**
  - Electrically controlled switches (more on this later)
  - Anyone have any idea how many transistors are in a modern chip? **2.6 BILLION**

- How do you put together 2.6 Billion of anything
  - ...and make sure the product works right?
  - ...in every corner case
  - ...and is really fast
  - ...and do it within a reasonable budget/timeframe?

- More fundamentally, how do you engineer any large system?
Transistors => Processors

- Processors are made of **transistors**
  - Electrically controlled switches (more on this later)
  - Anyone have any idea how many transistors are in a modern chip? **2.6 BILLION**

- How do you put together 2.6 Billion of anything
  - ...and make sure the product works right?
  - ...in every corner case
  - ...and is really fast
  - ...and do it within a reasonable budget/timeframe?

- More fundamentally, how do you engineer any large system? **Abstraction**
Abstraction: The Key to Computer Engineering

• Abstraction: Divide interface from implementation
  • Interface: how its used
  • Implementation: how it does it

• Build larger components from smaller ones
  • Larger ones use interface of smaller ones to do tasks
  • Don’t care about implementation

• Tasks can be split between engineers:
  • You make a piece that does xyz, and I’ll use it to do my job

• Components can be re-used
  • Also good: making them generic, so they can be re-used more
Other key to engineering: tools

- Processors designed in Hardware Design Languages
  - Verilog
  - VHDL
    - Learn one: you can pick the other easily

- You don’t lay out every transistor by hand...
  - Instead you write a description of the hardware in an HDL
  - ...a lot like a programming language...
  - ...then run it through synthesis tools

- We’ll use VHDL and Quartus
  - With ModelSim to simulate
  - This week’s recitation
Levels of Abstraction

• Transistors: “electrical switch”
  • Can go lower (those with EE background have)
  • ..but no need for us

• Gates: a few transistors
  • Implement logical functions: And, Or, Nor, Xor

• Meaningful logic elements: a handful of gates
  • Combine into meaningful elements: muxes, 1-bit adders, flip-flops
  • May build larger items: N-bit adders from 1-bit adders

• Large elements (stages, units): combining logic elements

• Core

• Chip: now with multiple cores
A Different Kind of Abstraction

• Previous discussion: abstraction to build a processor

• Also: abstraction to **use** a processor
  • How/why?
A Different Kind of Abstraction

• Previous discussion: abstraction to build a processor

• Also: abstraction to **use** a processor
  • How/why?
  • Need software that can use the processor
  • Software should not rely on **HOW** processor is implemented
  • Abstraction between hardware implementation and interface
    • Interface = ISA = contract between hardware and software
    • Implementation: can vary from generation to generation
    • Consider x86
      • Can take a program written for an i386 (1985)
      • ...and run it on a modern core in 2016
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 ECE550 tech tree

1. We have **transistors**. They’re electronic switches.
2. You can make **logic** by combining transistors.
3. You can make **arithmetic** from logic.
4. Transistors can also make basic **memory**.
5. Combine arithmetic and logic circuits to make an **Arithmetic/Logic Unit (ALU)**
6. Combine memory elements to make **registers** to store values
7. Combine the ALU, registers, and other stuff to make a full **CPU**!
8. Use other transistor configurations to make large-scale **RAM**.
9. RAM is too slow, so add **cache**, another form of memory to speed it up.
10. We have **storage** in the form of hard drives and SSDs.
11. Use storage and RAM together: this is **virtual memory**, which allows us to run more programs with more efficiency.
12. We halt the CPU with **exceptions** to handle things like **I/O** to storage.
13. The modern computer is governed by basic software called the **operating system**.
14. We have **networking** to connect multiple computers.
15. CPU performance is increased by various techniques, including **pipelining**.