EE 201: Fundamentals of computer systems and engineering

Spring 2019
T/Th 4:30-5:45pm, Halligan 108

Welcome to EE 201!

Any sufficiently advanced technology is indistinguishible from magic. – Arthur C. Clarke

There are few technologies for which Clarke's statement is more true than microprocessors. Today you can find a microprocessor in almost anything electronic, and yet even those of us who can program them often have very little understanding of their inner workings. In EE 201, we will study microprocessors — and digital electronics more generally — from the ground up. From your past courses, you know how computers work at the physical level (voltages and currents) and at the application level (C++ and other languages); the objective of this course is to connect the dots between the two.

To do this, we begin with the basics of manipulating zeros and ones with circuitry, and start building *combinational* circuits, which produce outputs based on some mathematical combination of their inputs. Then we make a major turn and explore *sequential* circuits, which can store information or step through a sequence of states. With these building blocks in hand, we will examine the ARM instruction set (which powers your cell phone and a few billion other devices), and build circuits which can actually interpret and execute software instructions. Finally, we'll discuss popular techniques to make processors fast, including pipelining (executing multiple instructions at once in assembly-line fashion) and caching (saving important data nearby for faster access).

When we're done, you'll have built a working microprocessor from scratch, and implemented it on an FPGA. You'll also be able to create hardware to search through genomes, analyze and manipulate network traffic, mine bitcoin, or play retro video games. In short, you'll be able to work magic.

After successfully completing this course you will be able to:

- Use modern digital development tools including HDL synthesis tools and logic analyzers to implement and debug combinational and sequential logic circuits on an FPGA.
- Describe how a microprocessor works in terms of intermediate digital building blocks. That is, you should be able to explain to someone halfway through the course how a pile of logic gates can read a sequence of instructions and operate on a block of data to produce results.
- Explain the purpose of pipelining and caching, analyze the performance of a system using pipelining and/or caching, and design the hardware components necessary to implement such a system.
- Use memory-mapped peripherals on a microcontroller, and design memory-mapped hardware for timing, general-purpose I/O, and SPI or I2C communication.

How will this help me?

With the skills from this course, you'll be able to design computer systems which can acheive a hundred or thousand times more performance than running code on a traditional CPU.

Even if you don't ever design digital hardware, a solid understanding of how the hardware works will enable you to write better code. You'll be able to read and understand those thousand-page microprocessor datasheets from Microchip and NXP, and even on desktop CPUs you may be able to achieve 10× faster on the same hardware. Moreover, specialized computer hardware is everywhere, from tensor processors in datacenters to neural compute engines in cell phones. These special processors are especially quirky, and only engineers who understand the hardware will be able to wring out the highest possible performance.

And "performance" isn't just speed: power efficiency is incredibly important for wearable and some IoT applications, and this too requires a solid knowledge of digital hardware, even if you're "just" writing code.

Perhaps most importantly, you'll have more tools in your toolbox: you'll know exactly what you can do with a CPU, what you can do with an FPGA, and when to use each of them.

Communication:

Steven Bell `sbell@ece.tufts.edu`

Halligan Hall 009 (the very back of the extension, at least until they knock it down)

Office hours:

- Monday 3-4pm (walking office hours, see my website for details)
- Wednesday 3-4:30pm
- Thursday 1-3pm
- I'm also available other times by appointment, or just drop by my office any time the door is open.

To minimize distraction, I generally only check email a few times a day. However, I will make a strong effort to answer all messages within 24 hours on weekdays.

All materials will be posted on the course website: http://www.ece.tufts.edu/ee/201. We will not use Canvas at all.

All assigment submissions and feedback will be done through Gradescope. We'll provide the signup link once the course gets started.

If you have a general question about the course content or course logistics, please post on Piazza rather than emailing the teaching staff. You'll usually get a faster response, and everyone benefits from the answer. Sign up for this course on Piazza here: http://piazza.com/tufts/spring2019/ee201.

Textbook:

Sarah L. Harris & David Money Harris, *Digital Design and Computer Architecture, ARM Edition*. ISBN: 978-0128000564

The textbook is required, since there will be regularly assigned readings and quizzes on what you've read.

There are some online resources for the book, including an electronic-only chapter 9: https://booksite.elsevier.com/9780128000564/

Prerequisites:

There are no prerequisites for this course except graduate standing. Some familiarity with C/C++ or assembly programming (such as COMP 40 or EE 200) will be helpful.

Exams:

There will be one in-class midterm exam, tentatively scheduled for March 7. The final exam will be on Monday, May 9 2019, 3:30-5:30pm (L block).

Grading:

Grades will be assigned on an absolute scale (not curved), with the following components:

Pre-class and in-class quizzes: 10 %

Homework: 50 %

Exams: 40 %

Homework:

The weekly homework assignments will released each Tuesday and due the following Tuesday. Homework will be a mix of traditional pen-and-paper problems, VHDL coding challenges where you build components of your microprocessor, and "lab" problems where you implement VHDL designs on an FPGA and wire up circuitry to support your design.

Late policy:

Life gets crazy during the semester, so you may use up to 4 late days to submit homework without penalty. You do not need to do anything special to use a late day; just submit your assignment on Gradescope as normal, and it will be marked late. For simplicity of bookkeeping, late days must be used in whole days; i.e., an assignment 5 minutes late counts for one late day, as does an assignment 23 hours late.

Assignments later than your late days but within a week of the due date may be submitted for half credit. (We'd rather you submit late than not at all, but we have to get on with grading eventually.)

Laptops and phones in class:

I aim to make our classroom an interactive and collaborative environment. To facilitate this, all "devices" (cell phones, laptops, tablets, Palm Pilots, Furbys, GameBoys, etc.) should be put away during class, except when you are asked to use your device for an online quiz or poll.

ADA accomodations:

If you need special accommodations (extra time on exams, larger type on handouts, etc.), please initiate the process with Tufts Student Accessibility Services (http://students.tufts.edu/student-accessibility-services/accessibility-policies-procedures). Accomodations cannot be granted retroactively, so please do this sooner rather than later.

			Topic	Reading	Homework
Jan 17	Thursday	1	Welcome and introduction		
Jan 22	Tuesday	2	Boolean equations, truth tables, and circuits	1.5, 2.1-2.2	
Jan 24	Thursday	3	Manipulating boolean equations & minimizing logic	2.3-2.7	
Jan 29	Tuesday	4	Multiplexers and FPGAs	2.8	HW 1 due
Jan 31	Thursday	5	Timing combinational logic	2.9	
Feb 05	Tuesday	6	VHDL for combinational logic	4.1-4.2	HW 2 due
Feb 07	Thursday	7	Testing and testbenches	4.3, 4.9	
Feb 12	Tuesday	8	Latches and flip-flops	3.1-3.3	HW 3 due
Feb 14	Thursday	9	Basic sequential circuits	4.4-4.5, 5.4	
Feb 19	Tuesday	10	State machines	3.4	HW 4 due
Feb 21	Thursday		*No class; Monday schedule due to holiday*		
Feb 26	Tuesday	11	Timing sequential logic	3.5-3.7	HW 5 due
Feb 28	Thursday	12	Testing sequential logic		
Mar 05	Tuesday	13	Snow day / review for exam		HW 6 due
Mar 07	Thursday	14	**Midterm exam**		
Mar 12	Tuesday	15	Adders and other combinational circuits	5.1-5.2	
Mar 14	Thursday	16	Memory: registers, RAM, and ROM	5.5	
Mar 19	Tuesday		*No class, spring break*		
Mar 21	Thursday		*No class, spring break*		
Mar 26	Tuesday	17	ARM assembly programming	6.1-6.3	HW 7 due
Mar 28	Thursday	18	From assembly to 0s and 1s	6.4-6.9	
Apr 02	Tuesday	19	Building a single-cycle processor	7.1-7.3	HW 8 due
Apr 04	Thursday	20	A more efficient multi-cycle processor	7.4	
Apr 09	Tuesday	21	Even faster: pipelined processors	7.5	HW 9 due
Apr 11	Thursday	22	Processor catch-up day		
Apr 16	Tuesday	23	Techniques for modern processors	7.7-7.8	HW 10 due
Apr 18	Thursday	24	Digital peripherals and protocols	9.1-9.3.4.1	
Apr 23	Tuesday	25	Caching	8.1-8.3	HW 11 due
Apr 25	Thursday	26	Virtual memory	8.4-8.5	
May 09	Thursday		**Final exam, 3:30-5:30pm**		