91.305 COMPUTER ARCHITECTURE

Contact

Prof. Fred G. Martin
`fredm@cs.uml.edu`
Olsen 208 (office)
Olsen 306 (lab)
978/934–1964 (office phone)

Schedule

Monday, Wednesday, Fridays, 10:30 am – 11:20 am, Olsen 414.

Office Hours / Lab Times

Schedule to be announced.

Course Web Site URL

`http://www.cs.uml.edu/~fredm/courses/91.305/`

Discussion Site URL

There will be a discussion site / bulletin board for the class. It will be linked from the course home page indicated above.

Course Description

From the UML CS brochure:

> "An examination of the basic functional components of a computer system including the CPU, memory systems, and I/O systems. Each of these three areas will be developed in detail with a focus on the system design and component integration. Topics will include CPU control and ALU operation, computer timing, data address and I/O bus activity, addressing model, programmed and DMA I/O, and instruction sets and microcode."

Now, my words. We will learn the structure of computers and how they work. We'll study computers as layered systems, like a Russian nesting doll. From the inside of the doll (lowest level) moving outward, here's one way to describe the layers:

0. semiconductor physics—electron flow; how transistors work

1. transistors as switches / logic gates / boolean operations—for our purposes, the most primitive element of a computational system

2. adders / math units / mutiplexers / decoders—the stuff you can build with gates, which become the building blocks of the CPU (central processing unit)

3. CPU—the heart of the computer. It's implemented by what's called the "microarchitecture level." Its external interface—the way you experience it as a programmer—is called the "instruction set archictecture" or machine language.

4. memory, peripherals [input/output], and the other doo-dads the CPU talks to—logically, this isn't actually above the CPU, but rather along side it

5. compilers—software that takes higher level code written by people and generates machine code to run on the CPU. The compiler matters because (1) if you're writing a compiler, you need to intimately understand the CPU to do a proper job, and more commonly (2) if you're using a compiler [who doesn't?] you need to understand how it works to write decent code.

6. operating system—in a sense the most important program the CPU will run. The OS doesn't do useful work by itself, but it implements a rich set of services that application programmers use.

In this class, we won't worry about my "level 0," and we probably won't get to anything too important in "level 6." For "level 0," see 16.423/16.523 Intro to Solid-State Physical Electronics; for "level 6," see 91.308, Introduction to Operating Systems !

Text

Structured Computer Organization, 4th Ed., by Andrew S. Tanenbaum (Prentice-Hall, 1999). The author is pretty famous, having also written landmark texts on computer networking and operating systems. Tanenbaum can also claim Linus Torvalds as a student, who after taking Tanenbaum's OS course, went on to write Linux! See `http://www.cs.vu.nl/~ast/home/faq.html` for more.

Strategy

This will be a lab class where you'll actually build stuff with chips and wires. The systems that we'll build will be simpler than the ones we'll read about and talk about. My intention is that the more complex stuff will be more approachable and relevant because we'll have had hands-on experience with more basic systems.

Grading

in-class participation, 10%
written assignments, 15%
lab assignments, 25%
mid-term exam, 25%
final exam, 25%

Policy

In professional as well as academic life outside the classroom, people seldom work completely on their own. They typically work in teams and help each other extensively. I have no objection to you getting help from me or your fellow students. I encourage you to do so. However, unless specifically stated otherwise by me, *prepared work in this course is to be each student's own*.

Students should therefore be familiar with the University's definitions and policies on academic dishonesty, found in the University course catalog. [above adapted from Prof. Jesse Heines' copying policy]

The contributions of others to your thinking must be acknowledged in all work you turn in. As UML Prof. Sarah Kuhn says, "*Using works of others, or drawing extensively on their ideas, without clearly stating that they are not your work (by using quotation marks, and references to the cited work) is plagiarism, a very serious academic offense.*" [Prof. Sarah Kuhn, syllabus for 65.790, from Prof. Marian William's 91.531 course syllabus.] With each assignment, you must mention people whom you worked with, who you have helped, or who have helped you.