Computer Organization

Course Syllabus[1] — Fall 2017

The focus of this course is on the selection and interconnection of components that make up a computer. There are two essential categories of components in modern computers: the hardware (the physical medium of computation) and the software (the instructions executed by the computer). As technology becomes more complex, the distinction between hardware and software blurs. We study why this happens, as well as why hardware designers need to be concerned with the way software designers write programs and vice versa. Along the way, we learn how computers work from higher-level programming languages such as Python and JavaScript, to system-level languages like C and Java, down to the basic zeroes and ones of machine code. Topics include Boolean logic, circuit design, computer arithmetic, assembly and machine languages, memory hierarchies, and parallel processing.

While we discuss some aspects of x86 architecture we will focus on RISC architectures including MIPS and the now-ubiquitous ARM. Special attention will be given to the recently-developed, open-source RISC-V architecture.

Permission of the instructor is required. Students should have at least one semester of programming experience, preferably in C, C++, Java or Python.

DRAFT SYLLABUS

Meeting times

Mondays and Wednedays, 9–10:55, Science 214

Instructor

Michael Siff
Faculty of Computer Science
Science Center 213
siff.michael@gmail.com
914/395-2490

Required text

Patterson, David A. and Hennessy, John L. *Computer Organization and Design RISC-V Edition: The Hardware Software Interface*. Morgan Kaufmann, 2018.

The book is available for purchase at the Sarah Lawrence Bookstore and is on reserve at the Esther Raushenbush Library. (There are many editions of this text book; the one we will be using has only very recently been published and is specific to the RISC-V architecture which we will be studying in class. However, if you only have access to another edition, aim for one of the MIPS-based editions preferably the most-recent Fifth Edition from 2013.)

[1] Draft syllabus as of August 25, 2017, for consideration during registration.

Homework

Readings. You are expected to keep up with assigned readings; they will not be long, but will be fairly technical which means it is best to pace yourself. It may also be useful to reread after we have discussed the material in class to reinforce specific concepts.

Daily exercises. In preparation for class, you will be asked to work out a few exercises, write a short programming fragment, or use design software to simulate a hardware component. These short assignments are not for formal submission, but I will call on volunteers to present their solutions in class. By doing these exercises in a timely fashion you will be more prepared for the day's lesson and that in turn will allow us to cover more material in a less hurried fashion.

Problem sets. There will be three problem sets. There will be two due dates associated with each problem set. By the start of class on the first date, you need to have seriously attempted each of the problems in the set. During that class we will discuss some of the problems and I may give hints to others. A clearly organized, typed set of solutions to the problems will be due on the latter date.

Programming assignments. There will be four programming assignments: one in a high-level language of your choice (a two-pass assembler); one in C (a disassembler); and two in RISC-V assembly (the Sieve of Eratosthenes and linked lists).

Circuit designs. There will two circuit-design assignment. The first will use JLS to simulate the feedback loop in a D flip-flop. The second will use Logisim to simulate both a register file and a bit shifter.

Design project. There will one substantial team-based circuit-design project due near the end of the semester. Each team will be expected to give an in-class presentation of their work.

DRAFT SYLLABUS

Problem sets, programs, and circuit designs, unless explicitly stated otherwise, *are to be completed independently* (see more under the "academic integrity" section).

Late submissions will not be accepted.

Conference

Conferences will be held every other week, unless you specifically request a weekly meeting. You are expected to come to the first conference armed with at least two ideas for conference projects. I am open to almost any computer-science related topic, although preferably one related to computer organization or architecture. Here are a few ideas you might think about:

- benchmarking
- compilers, interpreters and hardware
- graphics processing units (GPUs)
- hardware-description languages (e.g., Verilog)
- historical and/or economic survey of computer architectures
- input and output devices (and their evolution)
- lower-level circuit design & electronics
- virtual machines (e.g., JVM)
- x86 architecture
- ARM programming on Raspberry Pi
- computer hardware in pop culture

You may be asked to give a short (10–15 minute) presentation describing your conference work near the end of the semester.

DRAFT SYLLABUS

Attendance

As stated on p. 23 of the Student Handbook: "Consistent attendance at all classes and conferences is expected." Unexcused absences or persistent lateness, whether in class, lab, or group conference, may lead to reduced credit. In this era of ubiquitous connectivity, an absence for which a student did not inform the instructor of in advance is *unexcused*. (That is *not* to say that if merely informing in advance will excuse an absence, but it will often help.)

Checking email and the course Web page

Students are expected to regularly check their Sarah Lawrence (`@gm.slc.edu`) email account.

Academic integrity

Copying code, whether electronically (as in downloading, file sharing, copying and pasting) is plagiarism. Retyping someone else's solution is also plagiarism. Paraphrasing someone else's code by using different spacing and variables names is still plagiarism. Web searching (whether using Google, StackOverflow or other sources)

for solutions to assigned problems on line (for example, googling "c code to reverse array") is prohibited.

Please review the section entitled "Undergraduate Policy on Academic Integrity" in the Student Handbook (pp. 21–23). If you are at all unsure if your work habits are in keeping with the spirit of "working independently", be sure to check with me.

DRAFT SYLLABUS

Calendar

The following represents a *preliminary* schedule of topics and assignments.

Week	Date	Topic	Read	Due
1	9/04	introduction & overview		
	9/06	bits, binary, Boolean	A.1–3	
	9/08			prog 0 (2-pass assembler)
2	9/11	sum-of-products algorithm	A.5–6	
	9/13	feedback & sequential circuits	A.7–9	
3	9/18	assembly language intro	2.1–3	
	9/20	machine code	2.4–5	
	9/22			circuit 0 (JLS flip flop)
4	9/25	C primer I	*tba*	
	9/27	C primer II	*tba*	problem set 0
5	10/02	bitwise/logical operations	2.6	
	10/04	control structures	2.7	
	10/06			circuit 1 (register file/bit shifter)
6	10/09	C primer III	*tba*	
	10/11	C primer IV	*tba*	
7	10/16	*no class — October Study Days*		
	10/18	functions & recursion	2.8	
	10/20			prog 1 (disassembler in C)
8	10/23	strings & pointers	2.9	
	10/25	more fun with pointers	2.13–14	
9	10/30	computer arithmetic	3.1–2	
	11/01	multiplication	3.3	problem set 1
10	11/06	division	3.4	
	11/08	floating point	3.5	
	11/10			prog 2 (Sieve in RISC-V assembly)
11	11/13	basic ALU	4.1–2	
	11/15	single-cycle datapath	4.3–4	
12	11/20	*presentation of design projects*		
	11/22	*no class — Thanksgiving*		
13	11/27	memory hierarchy & cache	5.1–3	
	11/29	virtual memory	5.6–7	
14	12/04	pipelining	4.5–6	
	12/06	hazards & exceptions	4.7–9	
	12/08			prog 3 (linked lists in RISC-V)
15	12/11	parallelism I	4.10; 6.1–2	
	12/13	parallelism II	*tba*	problem set 2

DRAFT SYLLABUS