# CSE378 Fall 2007

Computer Organization and Assembly Programming

**Contact info:**


Luis Ceze, instructor, luisceze@cs
Joe Devietti, TA, devietti@cs
Anindita Mitra, TA, anin10@u
Shen-Hui Lee, SLA, shen@cs
Colin Bayer, SLA, vogon@cs

All of us at once: cse378-tas@cs.washington.edu
The entire class: cse378@cs.washington.edu


**Meetings:** Lectures MWF, 10:30-11:20, EEB 045
Section AA, Th, 9:30-10:20, MGH 254
Section AB, Th, 12:30-01:20, MGH 228

**Book:** David Patterson and John Hennessy, "Computer Organization and Design", 3rd ed. *A Note about international editions:* generally, purchasing and using the international version of this book is fine for this class. Homework assignments are given by the US edition, however, and the numbering for problems is different between the international and domestic editions. Furthermore, the domestic edition contains a helpful green card in the front. This green card will be your best friend in the lab. International editions contain the same card, but on thin paper. Nevertheless, its your money so it is up to you.

**Grading:** 35% Labs, 20% Homework, 15% Midterm, 25% Final, 5% Class participation.

**Due dates:** All deadlines are at 5pm on the date stated.

**Late policy:** Any assignment (except the midterm and final) can be turned in late without penalty as long as the *total* number of late days does not exceed 8. Exceeding 8 late days (totaled over all assignments) will decrease your grade by 10% per extra late day for the late assignment.

**Homework:** There are about 4 homework assignments in this class. You can expect that some will include programming assignments. The programming assignments will be mostly on writing programs in MIPS assembly and running them using a program called "SPIM", which is a MIPS simulator. You can download your own copy of SPIM and install it on your computers, or use the one pre-installed on the department machines. There will be book work assignments for good measure. Programming assignments that are not in MIPS assembly can be written in any high-level language.

**Labs:** This class has a significant lab component, and as such is a good portion of your grade. There are 4 labs. Finishing each lab is likely to require all the time we give for it. START THE LAB SOON AFTER WE HAND IT TO YOU. THEY MIGHT BE TIME CONSUMING. These labs will walk you through the design and construction of a pipelined MIPS processor. When you finish these labs you will have designed a fully working MIPS processor, capable of executing arbitrary C code, and one that runs on real hardware inside of an FPGA. It'll be fun, trust us. You should work on the lab in pairs. While you can work alone on it, given the *substantial* time commitment required, we *highly* recommend you work with a partner.

**Exams:** There will be a midterm and a final. We expect that if you attend class, keep up with reading, and are doing well in the lab, then the exams should be smooth.

**Class Wiki:** A Wiki has been set up for this class in order to facilitate communication between students and to give you a place to note down any interesting discoveries or problems that you have run into while working on the labs or assignments so that others can benefit from your experiences. The Wiki also contains wikified versions of the labs and assignments that you can edit if you feel that something is unclear about them. We will review modifications to the assignments and transfer them back to the actual pages if they are helpful. The wiki can be accessed through the Wiki link on the course website. Constructive participation on the wiki will count towards your class participation score.

**Cheating:** Don't go there; it'll end your academic career.

**Topics:**

- basic computer organization
  - CPU, memory, I/O
  - representation of data
- performance metrics for computer systems
  - execution time, CPI, MIPS, MFLOPS
- instruction set design
  - registers
  - arithmetic-logical instructions
  - load-store instructions and operand addressing
  - flow of control instructions
- instruction encoding
  - instruction formats
  - RISC vs. CISC
- translation of HLL program into assembly
  - register, user stack, static data area, heap
  - procedure call conventions
- how a program is compiled and executed
- processor implementation
  - datapath/control
  - pipelining
  - ideal pipeline
  - data hazards & forwarding
  - control hazards & branch prediction
- instruction-level parallelism
- memory hierarchy
  - caches, cache organizations
  - performance metrics for caches, taxonomy of cache misses
  - parameters for cache design
  - write strategies
  - virtual memory, page tables, TLBs
- exceptions/interrupts (interaction with operating system)