CS161: Design and Architecture of Computer Systems
Expanded Notes – Introduction and Logistics

University of California, Riverside

Instructor and Staff

This course is taught by Professor **Elaheh Sadredini**, an assistant professor in the CS department. She earned her PhD at the University of Virginia in 2019. Her research interests span hardware acceleration for big data (such as NLP, AI, and bioinformatics), energy-efficient cryptography at the edge, and secure, privacy-preserving computing. In short, she studies how we can design computing systems that are both fast and trustworthy.

Supporting her is the teaching assistant, **Sahar Ghoflsaz**, a PhD student whose work also touches secure computing and computer architecture. Grading is handled by designated graders; any concerns about scores should be directed to them first, with the instructor copied.

Course Logistics

Course information is spread across a few platforms:

- The **course website** hosts the syllabus, schedule, and policy details.
- **Canvas** is used for announcements, assignments, exams, and lecture slides.
- A dedicated **Slack** space supports discussion.

There are a few no-class days: October 14 (conference), November 11 (Veterans Day), and November 27 (Thanksgiving). Across the quarter there will be 18 lecture sessions.

Why Study Computer Architecture?

Computer architecture is about understanding the design layers that make modern computing possible. In the past, performance grew steadily under Moore's Law, but that scaling has slowed. Simply cranking up clock speed is no longer viable, since power and heat limits create a "power wall." Instead, designers turned to multi-core processors, specialized accelerators, and more energy-efficient designs.

For a programmer, knowing the hardware is valuable: it reveals why code behaves as it does, and what choices lead to efficiency. For those aiming to design chips, operating systems, or compilers, this knowledge is essential.

A central concept is the **Instruction Set Architecture (ISA)**. This is the interface between software and hardware—the layer that defines what instructions a processor can execute, how data is handled, and how control flow works. There are many ISAs in use: x86 (Intel/AMD), ARM (common in phones and laptops), PowerPC, and MIPS. This course uses **MIPS** because it is simple to learn, even though it is no longer a mainstream processor. Historically, MIPS powered graphics workstations, such as those used to create the effects in *Jurassic Park*.

What This Course Covers

The course develops understanding along four threads:

1. **Control**: how a MIPS computer runs programs at the lowest level, using its ISA and assembly instructions.
2. **Build**: how the datapath and control logic of a processor are put together at the circuit level.
3. **Performance**: how techniques like pipelining, branch prediction, caches, and parallelism squeeze efficiency out of hardware.
4. **Practical systems**: how real machines deal with memory management, input/output, and arithmetic.

The goal is not just abstract knowledge. It is to see how the decisions in architecture shape the speed, efficiency, and behavior of the systems we use daily.

Learning Objectives

By the end of the course, you should be able to:

- Describe how processors, memory, and I/O devices function and interact.
- Reason about performance quantitatively at a first-order level.
- Understand the hardware/software boundary and how it constrains both sides.

Evaluation and Workload

Grades are distributed as follows:

- 30% Homework assignments (HW0 is 2 points, HW1–HW8 are 4 points each, with one dropped).
- 40% Lab assignments.
- 20% Mini-teaching videos, where you will explain a concept.
- 10% In-class polling questions, graded by participation (answering 90% yields full credit).

This balance emphasizes both practice (labs and homework) and communication (teaching videos), with smaller weight given to participation.

Policies

Deadlines are strict. Homework submitted late is penalized 5% per hour, up to 10 hours. One homework may be skipped or dropped. Files that cannot be opened, or the wrong file, will not be accepted. Disputes over grading must be raised within one week.

Academic integrity is taken seriously. All work must be your own unless collaboration is explicitly permitted. Misconduct is reported to SCAIP. AI tools like ChatGPT can be used for learning or exploring, but cannot generate assignments for submission. Submitting AI-written work counts as cheating.

Resources

The optional textbook is Patterson and Hennessy's *Computer Organization and Design, MIPS Edition*, a classic in the field. Additional resources include lecture slides, online solutions, and TA-led discussions. Collaboration is encouraged, but each student's submission must reflect their own work.

First Week Tasks

At the start of the quarter:

- Complete the self-introduction task (worth 2 points).
- Review the basics of logic design to refresh the background needed for the course.