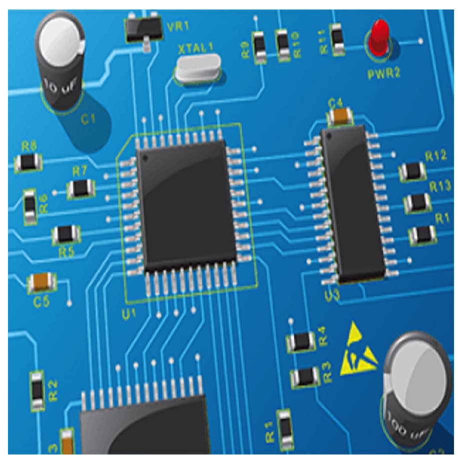
CSCI 341: Computer Organization

Spring 2026

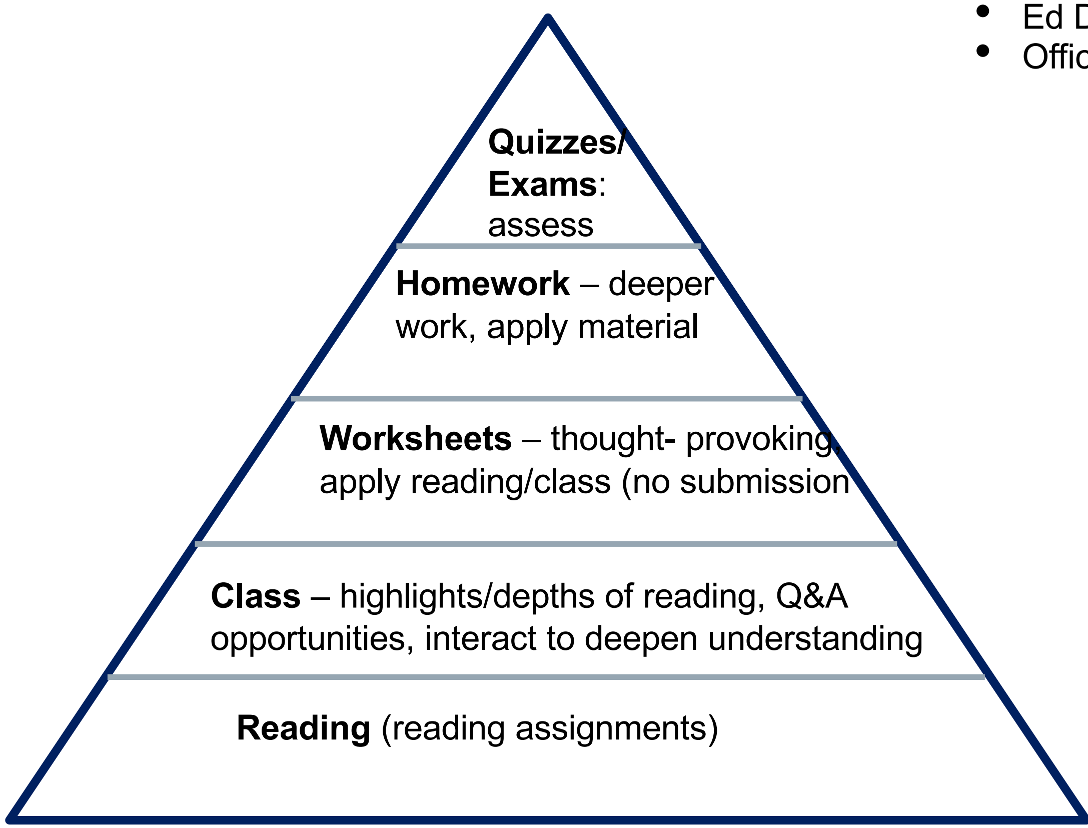
Dr. Qi Han



Topics for Module 1

- ➔ **Part 1: Course Logistics**
  - Syllabus
    - Teaching staff introduction
    - Course policies
    - Course workload
    - Canvas resources
  - Computer Abstractions and Technology (Chapters 1.1 – 1.5, 1.7)
- **Part 2: Review number representation in computer**
  - Fixed Point Number (i.e., integer) Representation
  - Floating Point Number Representation

Course Logistics



Resources:
- Textbook
- Canvas
- Ed Discussion
- Office hours (instructors & TAs)

Canvas:
- Syllabus
- Schedule & Policies
- HW/WS links
- Handouts

Office Hours:
- If available hours don't work, schedule an appointment with your instructor/TA

Ed Discussion:
- Announcements
- Help outside of OH
- Help your peers with concepts

What to Learn?

- How to represent numbers in computer
- How to assess and understand computer performance
- How computers work
  - **Instruction set architecture, Assembly Programming**
  - **Computer arithmetic**
  - **Processor design**
- Issues affecting modern processors (caches, pipelines)
  - **Pipelining – processor performance improvement**
  - **Memory system**

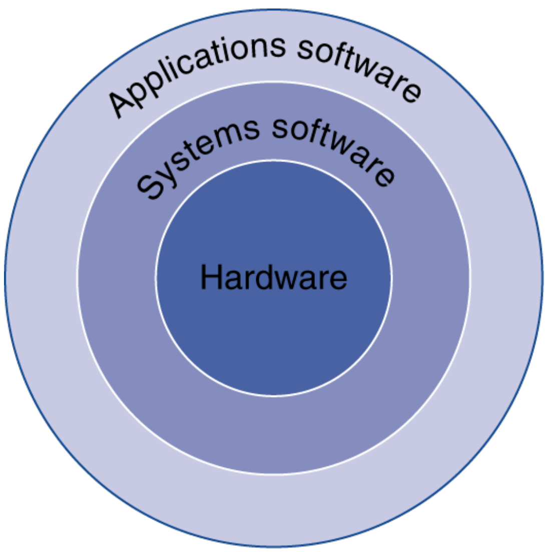
Why to Learn?

- You want to be a better programmer
- You need to make a purchasing decision or offer “expert” advice
- You want to call yourself a “computer scientist”

How to Learn?

- **Focus on a specific instance (RISC-V) and learn how it works.**
- **Why RISC-V instead of Intel 80x86 (CISC)?**
  - Simple, elegant, modern
  - Open instruction set
  - Open-source simulators, compilers, debuggers, etc.
  - Low-cost boards based on RISC-V available
  - Widespread commercial adoption across industries and implementations, from embedded automotive to hyperscale AI, from 5G to HPC and beyond

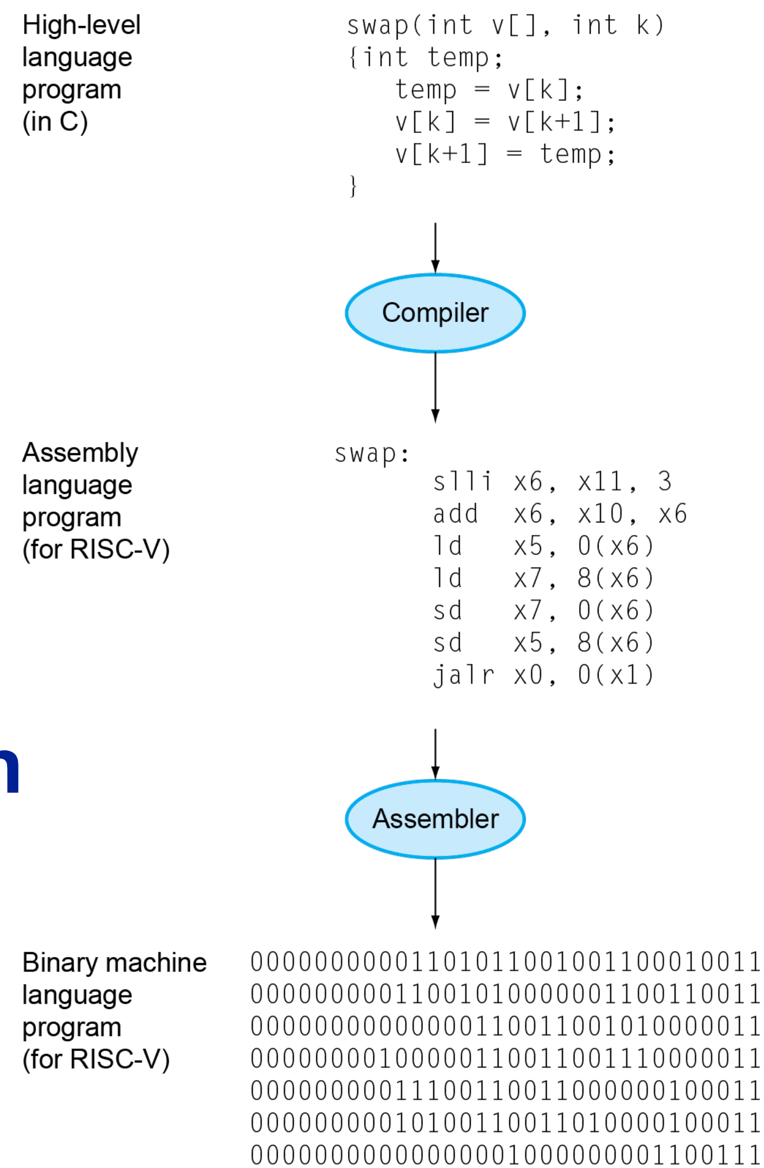
Below Your Program



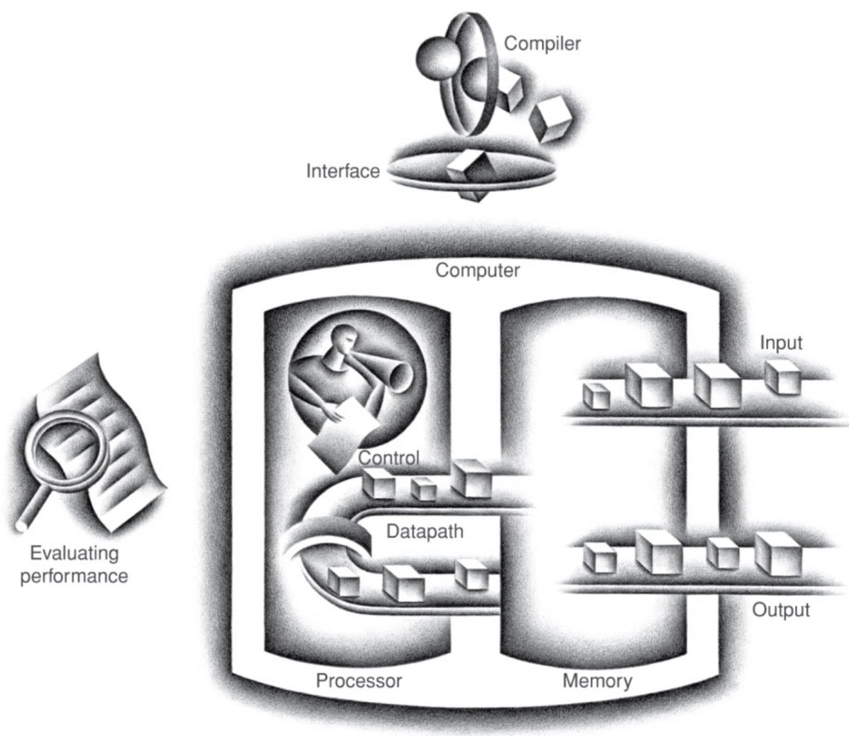
- **Application software**
  - Written in high-level language
- **System software**
  - Compiler: translates HLL code to machine code
  - Operating System: service code
    - Handling input/output
    - Managing memory and storage
    - Scheduling tasks & sharing resources
- **Hardware**
  - Processor, memory, I/O controllers

Levels of Program Code

- **High-level language**
  - Level of abstraction closer to problem domain
  - Provides for productivity and portability
- **Assembly language**
  - Textual representation of instructions
- **Hardware representation**
  - Binary digits (bits)
  - Encoded instructions and data



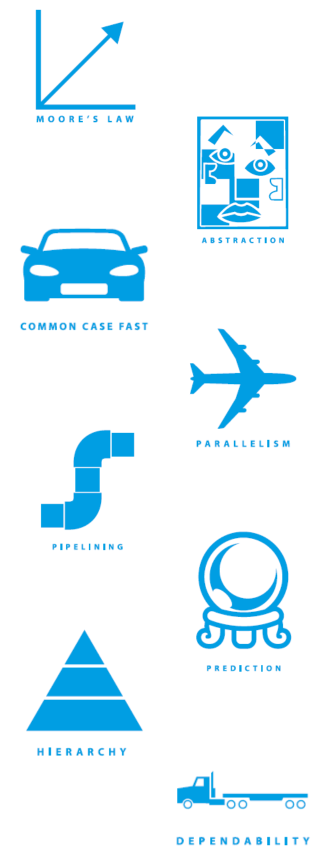
Components of a Computer



- **Same components for all kinds of computers**
  - Desktop, server, embedded
- **Processor**
- **Memory**
- **Input/output includes**
  - User-interface devices
    - Display, keyboard, mouse
  - Storage devices
    - Hard disk, CD/DVD, flash
  - Network adapters

Eight Great Ideas

- **Design for *Moore's Law***
- **Use *abstraction* to simplify design**
- **Make the *common case fast***
- **Performance *via parallelism***
- **Performance *via pipelining***
- **Performance *via prediction***
- ***Hierarchy* of memories**
- ***Dependability via* redundancy**



Inside the Processor (CPU)

- **Datapath: performs operations on data**
- **Control: sequences datapath, memory, ...**
- **Cache memory**
  - Small fast SRAM memory for immediate access to data