Introduction

This course is all about how computers work

But what do we mean by a computer?

- Different types: desktop, servers, embedded devices
- Different uses: automobiles, graphics, finance, genomics…
- Different manufacturers: Intel, Apple, IBM, Microsoft, Sun…
- Different underlying technologies and different costs!
Introduction

- Analogy: Consider a course on “automotive vehicles”
  - Many similarities from vehicle to vehicle (e.g., wheels)
  - Huge differences from vehicle to vehicle (e.g., gas vs. electric)

- Best way to learn:
  - Focus on a specific instance and learn how it works
  - While learning general principles and historical perspectives
Classes of Computing Apps.

- Desktop computers
- Servers
- Super computers
- Embedded computers
What to learn

- How are programs written in a high-level language, translated into the language of the hardware?

- What is the interface between the software and the hardware, and how does software instruct the hardware to perform needed function?

- What determines the performance of a program?

- What techniques can be used by hardware designers to improve performance?
Both Hardware and Software affect performance:

- Algorithm determines number of source-level statements
- Language/Compiler/Architecture determine machine instructions  
  (Chapter 2 and 3)
- Processor/Memory determine how fast instructions are executed  
  (Chapter 5, 6, and 7)
Below the application
What is a computer?

Components:
- input (mouse, keyboard)
- output (display, printer)
- memory (disk drives, DRAM, SRAM, CD)
- network

Our primary focus: the processor (datapath and control)
- implemented using millions of transistors
- Impossible to understand by looking at each transistor
Abstraction

High-level language program (in C)

swap(int v[], int k)
{
  int temp;
  temp = v[k];
  v[k] = v[k+1];
  v[k+1] = temp;
}

Compiler

Assembly language program (for MIPS)

swap:
  mli $2, $5, 4
  add $2, $4, $2
  lw $15, 0($2)
  lw $16, 4($2)
  lw $16, 0($2)
  sw $15, 4($2)
  jr $31

Assembler

Binary machine language program (for MIPS)

0000000001010000100000000000000000010000
000000000000000001000000000000000000001
1000110001100010000000000000000000000000
1000110011110010000000000000000000010000
1010110011111001000000000000000000000000
1010110011001000000000000000000000000000
00000011111000000000000000000000000001000
How do computers work?

Need to understand abstractions such as:

- Applications software
- Systems software
- Assembly Language
- Machine Language
- Architectural Issues: i.e., Caches, Virtual Memory, Pipelining
- Sequential logic, finite state machines
- Combinational logic, arithmetic circuits
- Boolean logic, 1s and 0s
- Transistors used to build logic gates (CMOS)
- Semiconductors/Silicon used to build transistors

Properties of atoms, electrons, and quantum dynamics
Instruction Set Architecture

- **A very important abstraction**
  - interface between hardware and low-level software
  - standardizes instructions, machine language bit patterns, etc.
  - advantage: *different implementations of the same architecture*
  - disadvantage: *sometimes prevents using new innovations*

*True or False: Binary compatibility is extraordinarily important?*

- **Modern instruction set architectures:**
  - IA-32, PowerPC, MIPS, SPARC, ARM, and others
Historical Perspective

- ENIAC built in World War II was the first general purpose computer
  - Used for computing artillery firing tables
  - 80 feet long by 8.5 feet high and several feet wide
  - Each of the twenty 10 digit registers was 2 feet long
  - Used 18,000 vacuum tubes
  - Performed 1900 additions per second
Eniac

–Since then:

Moore’s Law:

transistor capacity doubles every 18-24 months
Reading Assignment

- Read Ch 1. Especially 1.4 Performance