

Course Introduction; Chapter 1

- Rapidly changing field:
 - Vacuum tube -> transistor -> IC -> VLSI
 - Doubling every 1.5 years:
 - Memory capacity.
 - Processor Performance (Due to advances in technology and organization).
- Things you will be learning:
 - Representing numbers (and other things) in binary form.
 - How to write assembly language programs.
 - How computers work, a basic foundation.
 - How to analyze their performance (and how not to!).
 - Issues affecting modern processors (caches, pipelines) — if time permits.

Why learn this stuff?

- It's a required course :-)
- Useful in understanding other topics in CS.
 - CSc 352 — Systems Programming and Unix.
 - CSc 452 — Principles of Operating Systems.
 - CSc 453 — Compilers and Systems Software.
 - CSc 425 — Principles of Networking.
- You want to build software people use (need performance).
- You need to make a purchasing decision or offer “expert” advice.
- Learn the lingo (cache, MBytes, GBytes, MHz, GHz, pipelining, super-scalar, vector, etc.)

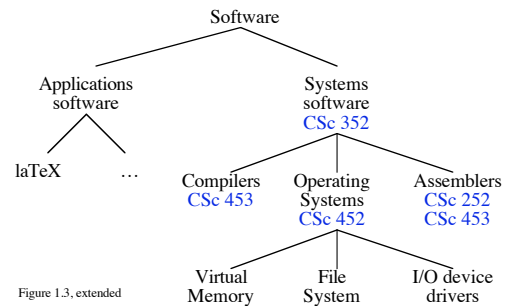


Figure 1.3, extended

What is a Computer?

- Components:
 - Input (mouse, keyboard, joystick, ...).
 - Output (display, printer, ...).
 - Memory (disk drives, DRAM, SRAM, SDRAM, DDR Ram, CD, ...).
 - Network.

- Our primary focus: the processor (Central Processing Unit, CPU).
 - Datapath and control.
 - Implemented using hundreds of millions of transistors.
 - Impossible to understand by looking at each transistor.
 - We need:

Abstraction:

- Delving into the depths reveals more information.
- Abstractions omit unneeded detail.

- Help us cope with complexity.
- What are some of the details that appear in these familiar abstractions?

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;
} /* swap */

```

C Compiler

Assembly
language
program
(for MIPS)

```

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

```

Assembler

Binary machine
language program
(for MIPS)

```

0000000010100001000000000000011000
00000000100011100001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
0000001111100000000000000001000

```

Figure 1.4, page 14

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:
 - 80x86/PentiumXX, Athlon/Duron, PowerPC, MIPS, UltraSparc, Itanium, Opteron, ...

Where We Are Headed:

- Binary representation of numbers and other things (Chapter 3).
- A specific instruction set architecture (Chapter 2 and Appendix A).
 - Introduction to assembly language programming.
 - Integer arithmetic.
 - Flow control.
 - Procedure calls.
- Arithmetic and how to build an ALU (Appendix B).
 - Data representations.
 - Floating point.
- Constructing a processor to execute our instructions (Chapter 5).
- Pipelining to improve performance (Chapter 6).
- Caching to improve memory performance (Chapter 7).

How to Do Well in this Course:

- Study the lecture notes (both those I hand out and those you add during class). Ask questions about what you do not understand.
- Read the text. Ask questions about what you do not understand.
- Work out the assignments (programs & homework assignments). Do so early! Ask questions about what you do not understand.
- Review returned quizzes, homework assignments, and programs. Ask questions about what you do not understand.
- Sign-up for the **comporg** listserv (see the syllabus).

Episode 7: New Allies, New Enemies
Star Wars - V: The Empire Strikes Back
NPR Original Radio Drama