Introduction to Computing Systems:

From Bits and Gates to C and Beyond

2nd Edition

Yale N. Patt
Sanjay J. Patel

Original slides from Gregory Byrd, North Carolina State University

Modified slides by Chris Wilcox, Andres Calderon J., Sanjay Rajopadhye, CSU

Lecture Goals

- Review course logistics
  - Assignments
  - Policies
  - Organization
  - Grading Criteria
- Introduce key concepts
  - Role of Abstraction
  - Software versus Hardware
  - Universal Computing Devices
  - Layered Model of Computing

Logistics

- Lectures: See syllabus
- Staff: See syllabus
- Recitations: See syllabus
- Help desks: See syllabus
- Office hours: See syllabus
- Materials on the website:
  - http://www.cs.colostate.edu/~cs270
- Piazza: access through Canvas

Assignments

Assignments are posted on website:

- Weekly assignments (mostly) alternate between written and programming assignments.
- Homework assignments: submission mode and deadline varies.
- Programming assignments are submitted in electronic form Sun. at 10pm.
- Late submission varies depending on the difficulty of the assignment.
- Regrading: through Piazza (see syllabus).

Policies

- Grading Criteria
  - Assignments (35%)
  - Recitations (10%)
  - Peer Instruction (5%)
  - Two Midterm Exams (15% each)
  - Final Exam (20%)
  - You must earn a passing grade (60% or higher) on each part – assignments and exams– in order to pass the class
- Late Policy
  - On-time = full points, late submission= 20% penalty
- Academic Integrity
  - http://www.cs.colostate.edu/~info/student-info.html
  - Do your own work
  - Be smart about Internet resources

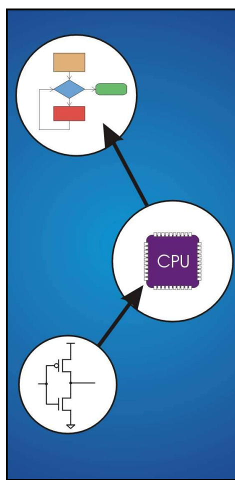
Organization

- 1/3 computer hardware: numbers and bits, transistors, gates, digital logic, state machines, von Neumann model, instruction sets, LC-3 architecture
- 1/3 assembly code: instruction formats, branching and control, LC-3 programming, subroutines, memory model (stack)
- 1/3 C programming: data types, language syntax, variables and operators, control structures, functions, pointers and arrays, memory model, recursion, I/O, data structures

Grading Criteria

How to be successful in this class:

1) Attend all classes and recitations, info will presented that you can't get anywhere else.
2) Do all the homework assignments, ask questions (early! (but not *too* early)) if you run into trouble.
3) Take advantage of lab sessions where help is available from instructors.
4) Read the textbook, work through the end of chapter problems.



Chapter 1 **Welcome Aboard**

Introduction to the World of Computing

- Computer: electronic genius?
  - NO! **Electronic idiot!**
  - Does exactly what we tell it to, nothing more.
- Goal of the course:
  - **You will be able to write programs in C**
  - **You will understand how a computer works (what's going on under the hood).**
- Textbook Approach:
  - From the bottom up (we will use mostly a top-down approach).
  - **Bits ➨ Transistors ➨ Gates ➨ Logic ➨ Processor ➨ Instructions ➨ Assembly Code ➨ C Programming**

Two Recurring Themes

- **Abstraction**
  - Productivity enhancer – don't need to worry about details…

    Can drive a car without knowing how the internal combustion engine works.
  - …until something goes wrong!

    Where's the dipstick?

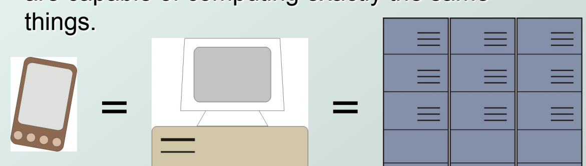
    What's a spark plug?
  - Important to understand the components and how they work together.

Two Recurring Themes

- **Hardware vs. Software**
  - It's not either/or – both are components of a computer system that cooperate.
  - Even if you specialize in one, you should understand capabilities and limitations of both.
  - The best programmers understand the computer systems which run their programs.
  - Computers are an entire ecosystem with multiple levels of abstraction.

Big Idea #1: Universal Computing Devices

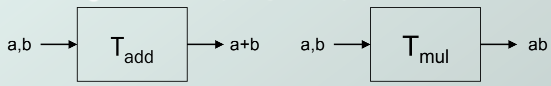
- All computers, given enough time and memory, are capable of computing exactly the same things.



PDA = Workstation = Supercomputer

Turing Machine

- Mathematical model of a device that can perform any computation – Alan Turing (1937)
  - ability to read/write symbols on an infinite "tape"
  - state transitions, based on current state and symbol
- Every computation can be performed by some Turing machine. *(Turing's thesis)*



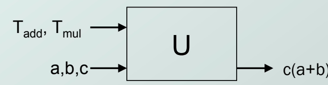
Turing machine that adds *Turing machine that multiplies*

For more info about Turing machines, see http://www.wikipedia.org/wiki/Turing_machine/

For more about Alan Turing, see http://www.turing.org.uk/turing/

Universal Turing Machine

- A machine that can implement all Turing machines – this is also a Turing machine!
  - inputs: data, description of computation (other TMs)



Universal Turing Machine

Universal machine is programmable – so is a computer!

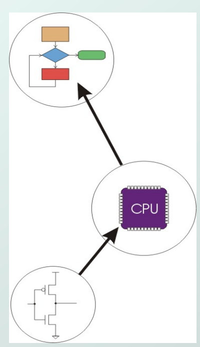
- **instructions are part of the input data**
- **a computer can emulate a Universal Turing Machine**

A computer is a universal computing device.

From Theory to Practice

- In theory, computer can ***compute*** anything that can possibly be computed
  - given enough *memory* and *time*
- In practice, ***solving problems*** involves computing under constraints.
  - time
    - weather forecast, next frame of animation, ...
  - cost
    - cell phone, automotive engine controller, ...
  - Power/energy
    - cell phone, handheld video game, ...

Big Idea #2: Transformations Between Layers



Problems

Algorithms

Language

Instruction Set Architecture

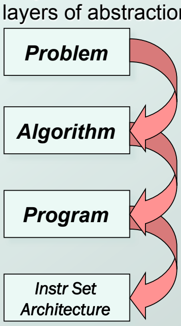
Microarchitecture

Circuits

Devices

How do we solve a problem using a computer?

- A systematic sequence of transformations between layers of abstraction.

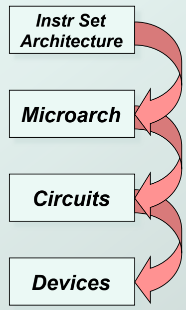


Software Design:
choose algorithms and data structures

Programming:
use language to express design

Compiling/Interpreting:
convert language to machine instructions

Deeper and Deeper…



Processor Design:
choose structures to implement ISA

Logic/Circuit Design:
gates and low-level circuits to implement components

Process Engineering & Fabrication:
develop and manufacture lowest-level components

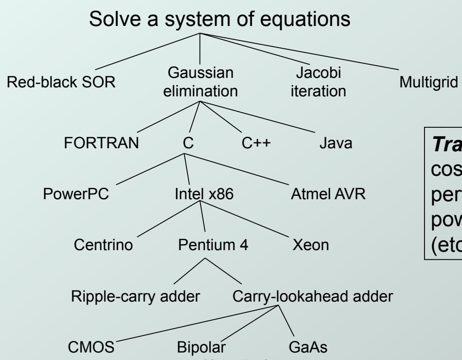
Descriptions of Each Level

- Problem Statement
  - stated using "natural language"
  - may be ambiguous, imprecise
- Algorithm
  - step-by-step procedure, guaranteed to finish
  - definiteness, effective computability, finiteness
- Program
  - express the algorithm using a computer language
  - high-level language, low-level language
- Instruction Set Architecture (ISA)
  - specifies the set of instructions the computer can perform
  - data types, addressing mode

Descriptions of Each Level (cont.)

- Microarchitecture
  - detailed organization of a processor implementation
  - different implementations of a single ISA
- Logic Circuits
  - combine basic operations to realize microarchitecture
  - many different ways to implement a single function (e.g., addition)
- Devices
  - properties of materials, manufacturability

Many Choices at Each Level



Tradeoffs:
cost
performance
power
(etc.)

Book Outline

- Bits and Bytes
  - How do we represent information using electrical signals?
- Digital Logic
  - How do we build circuits to process information?
- Processor and Instruction Set
  - How do we build a processor out of logic elements?
  - What operations (instructions) will we implement?
- Assembly Language Programming
  - How do we use processor instructions to implement algorithms?
  - How do we write modular, reusable code? (subroutines)
- I/O, Traps, and Interrupts
  - How does processor communicate with outside world?
- C Programming
  - How do we write programs in C?

Course Outline

- First, C programming (plus Bits/Bytes/Numbers)
  - Since you already have two semesters of Java
  - Learn the C memory model
  - How function parameters are passed (activation records, stack)
- Assembly Language Programming
  - How do we use processor instructions (three address instructions) to implement C programs (translation)?
  - How do we implement modular, reusable code? (subroutines)
  - How structured programs are broken down into “straight-line” code with (conditional) branches?
- Instruction set processor
  - Instructions and Data (the von Neumann model)?
- Digital circuits
  - Transistors and Gates, Memory and State machines
  - How the processor is built out of these (Register Transfer Notation)
- I/O, Traps, and Interrupts
  - How does processor communicate with outside world?