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 by Chris Wilcox, S. Rajopadhye Colorado State University

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: Tue and Thu from 12:30-1:45 in TILT 221
- Recitations (50 mins): at Wed 9, 11, 12, and Thu 2, 3 in CSB 225
- Exams:
  - One midterm in class (1 page notes, no calculator, no electronic aid)
  - final on Wednesday December 19 @ 6:20-8:20 pm in class (same rules)
- Materials on the website and RamCT:
  - http://www.cs.colostate.edu/~cs270
  - http://ramct.colostate.edu

Name cards

- Fold your card LENGTHWISE so that it can be placed on your desk
- Neatly write your name (how you want to be addressed) in BIG, BOLD LETTERS, using the markers that are circulating
- Prop it up on your desk so that it can be seen from the front of the classroom

Instructor

- Name: Sanjay Rajopadhye
- Pronunciation (optional): In Indian names, “a” is almost always pronounced as a short “u” sound as in gun, fun, etc., or a long “aa” sound as in calm, bard, etc.
- Sanjay is pronounced as Sun-juy
- Rajopadhye Raaj-Oh-paath-yay (in the “paath” make the t sound like a d). Don’t worry if you don’t get it right, it’s almost always mispronounced, even in India).
- These pronunciation rules are used in many parts of Asia, e.g., pronounce “Bagdad?”
- Interesting fact: Everyone in my family was born in a different country

TeachingAssistants (1)

- Name: Martin Muggli (muggli@rams.colostate.edu)
- Call me: Martin
- Pronunciation (optional):
- Major: Computer Science (Ph.D., 1st year)
- Interesting fact: Learning to play the cello as an adult

TeachingAssistants (2)

- Name: Sridhar Reddy Shyamala (shyamala@cs.colostate.edu)
- Call me: Sridhar
- Pronunciation (optional):
- Major: Computer Science (M.S., 1st year)
- Interesting fact: Loves Questioning

Assignments

Assignments and quizzes are posted on RamCT:

- Weekly assignments combination of written and programming assignments
- Written (hardcopy) assignments are due Thursday at 12:30 pm (***start*** of class)
- Programming assignments are submitted in electronic form Fridays at 12:00 noon.
- Reading Quizzes are online, the due date is Monday at 11:59pm

Policies

- Grading Criteria
  - Homework Assignments (45%)
  - Reading Quizzes (5%)
  - Recitations (5%)
  - Midterm Exam (20%)
  - Final Exam (25%)
- Late Policy
  - Accepted up to 24 hours after posted date with 10% penalty.
  - Not accepted after 24 hr delay
- Academic Integrity
  - http://www.cs.colostate.edu/~info/student-info.html
  - Be smart about Internet resources

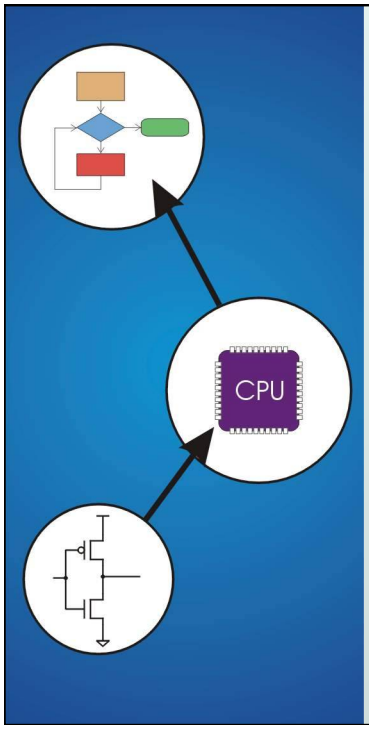
Organization

- 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, control, LC-3 programming, I/O, subroutines, memory model
- 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, information will presented that you can't get anywhere else.
2) Do all the homework assignments, ask questions (early!) if you run into trouble.
3) Read the textbook, take the quizzes, work through the end of chapter problems.
4) Use office hours



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 understand how computers are built, and write programs in C and understand what's going on underneath.
- Approach:
  - Build understanding from the bottom up.
  - **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?

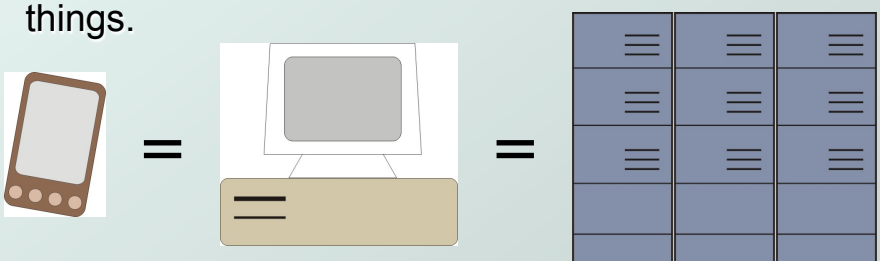
    What's a spark plug?
  - Important to understand the components and how they work together.

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

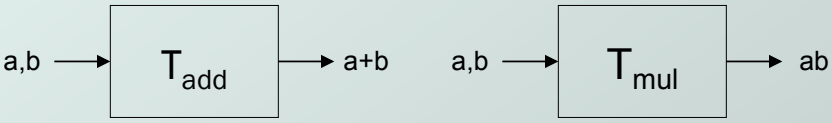
- All computers, given enough time and memory, are capable of computing exactly the same things.



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!

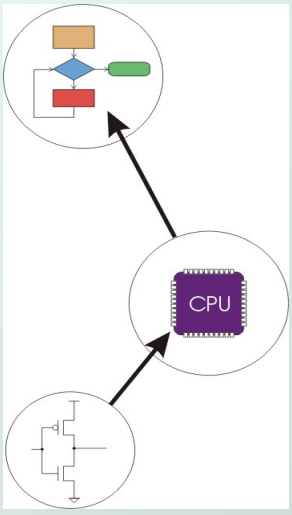
- **instructions are part of the input data**
- **a computer can emulate a Universal Turing Machine**

A computer is a universal computing device.

From Theory to Practice

- In theory, computer can ***compute*** anything
- that's possible to compute
  - 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
    - cell phone, handheld video game, ...

Big Idea #2: Transformations Between Layers



Problems

Algorithms

Language

Instruction Set Architecture

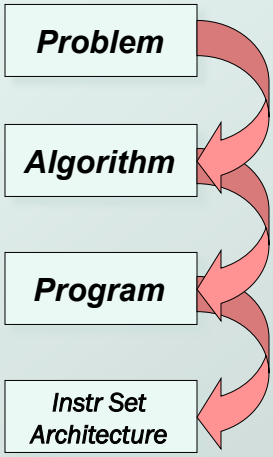
Microarchitecture

Circuits

Devices

How do we solve a problem using a computer?

- A systematic sequence of transformations between layers of abstraction.

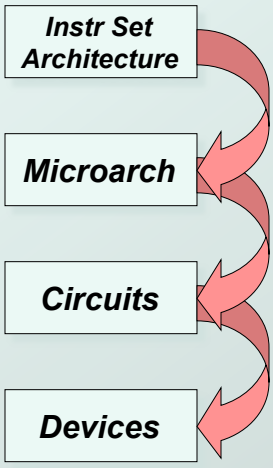


Software Design:
choose algorithms and data structures

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

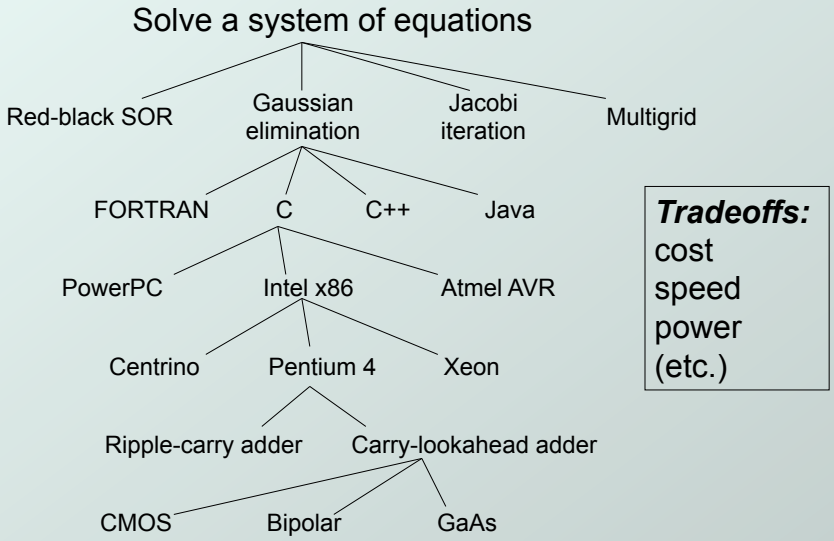
Descriptions of Each Level

- **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
speed
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?

Our Outline (CSU tweak)

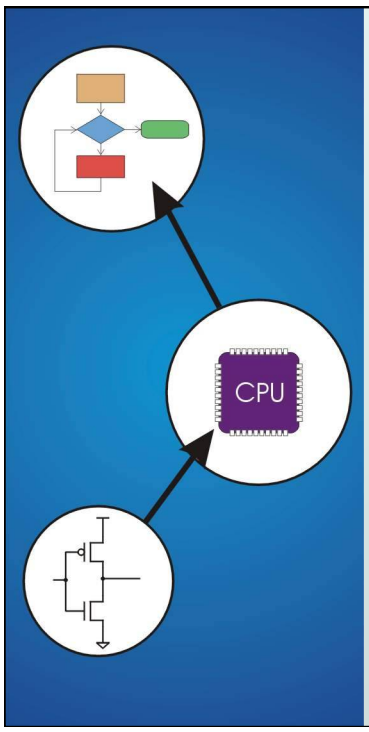
- First C programming
  - You already know how to program in Java
  - Learning C will be much faster than the book's speed (Ch 11-13)
- Follow the text sequentially
  - Digital Logic
  - Processor & Instruction Set
  - Assembly Language Programming
  - I/O, Traps, Interrupts
  - Functions, procedures, activations/frames, etc.
- Write a LC3 simulator and LC3 assembler in C
  - Reinforcement of ideas
  - Second look at the same ideas

How to succeed in this class

1) Most university classes require two to three times the contact hours.
2) So a 15 credit student should spend 60 hours in school related activity.
   1) Yeah right!!!
   2) **BUT IT'S TRUE ABOUT THIS CLASS**
3) Time management:
   1) 30 mins – Quiz
   2) 4 hrs – contact hours
   3) 30 mins – review class notes (15 x 2)
   4) 8-11 hrs – assignments

Before next lecture

1) Go to the class web page → Schedule tab
2) Download the "Number Representation" notes and study it
3) Spend one hour on it and do the exercises



Chapter 2
Bits, Data Types, and Operations

How do we represent data in a computer?

- At the lowest level, a computer is an electronic machine.
  - works by controlling the flow of electrons
- Easy to recognize two conditions:
  1. presence of a voltage – we'll call this state "1"
  2. absence of a voltage – we'll call this state "0"
- Could base state on *value* of voltage, but control and detection circuits more complex.
  - compare turning on a light switch to measuring or regulating voltage

Computer is a binary digital system.

Digital system:
- finite number of symbols

Binary (base two) system:
- has two states: 0 and 1

- Basic unit of information is the *binary digit*, or *bit*.
- Values with >2 states require multiple bits.
  - A collection of **two** bits has **four** possible states: **00, 01, 10, 11**
  - A collection of **three** bits has **eight** possible states: **000, 001, 010, 011, 100, 101, 110, 111**
  - *A collection of **n** bits has **2^n** possible states.*

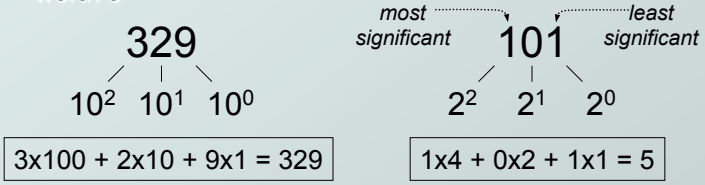
What kinds of data do we need to represent?

- **Numbers** – signed, unsigned, integers, floating point, complex, rational, irrational, …
- **Text** – characters, strings, …
- **Logical** – true, false
- **Images** – pixels, colors, shapes, …
- **Sound** – waveforms
- **Instructions**
- …

- Data type:
  - *representation* and *operations* within the computer
- We'll start with numbers…

Unsigned Integers

- Non-positional notation
  - could represent a number ("5") with a string of ones ("11111")
  - problems?
- Weighted positional notation
  - like decimal numbers: "329"
  - "3" is worth 300, because of its position, while "9" is only worth 9



Seen on a geek T Shirt

- There are only 10 kinds of people
- Those that know binary
- Those that don't