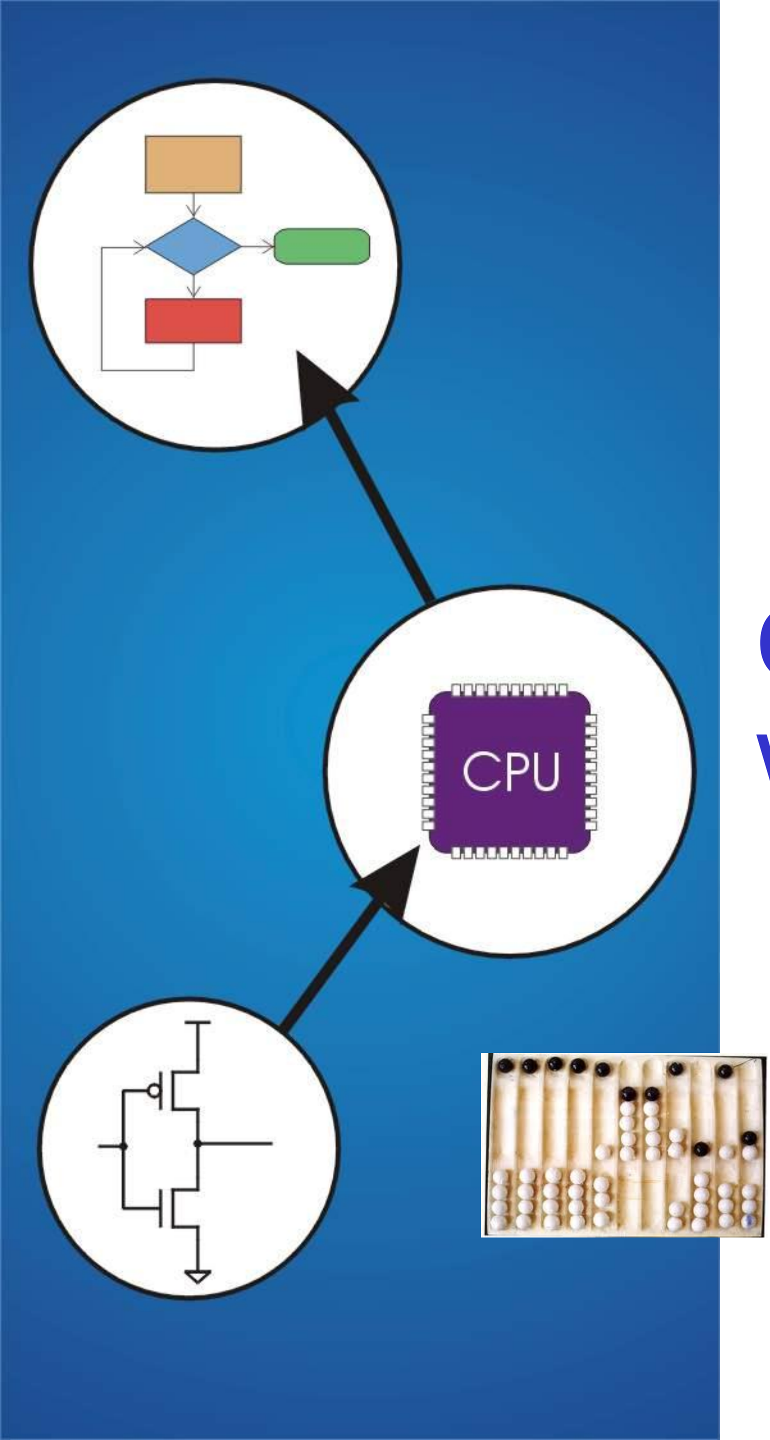
Introduction to Computing Systems:

From Bits and Gates to C and Beyond

2nd Edition

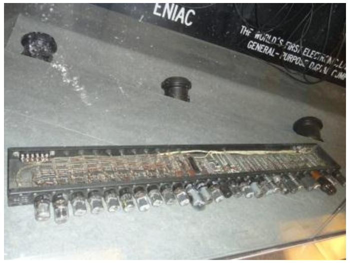
Yale N. Patt
Sanjay J. Patel

Slides prepared by
Gregory T. Byrd, North Carolina State University



Chapter 1
Welcome Aboard

ENIAC
THE WORLD'S FIRST ELECTRONIC LARGE SCALE GENERAL-PURPOSE DIGITAL COMPUTER


ENIAC
THE WORLD'S FIRST ELECTRONIC LARGE SCALE GENERAL-PURPOSE DIGITAL COMPUTER

Introduction to the World of Computing

Computer: electronic genius?

- NO! Electronic idiot!
- Does exactly what we tell it to do, nothing more.

Goal of the course:

You will be able to write programs in assembly language and understand what's going on underneath.

Approach:

Build understanding from the bottom up.

Bits ➔ Gates ➔ Processor ➔ Instructions

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?

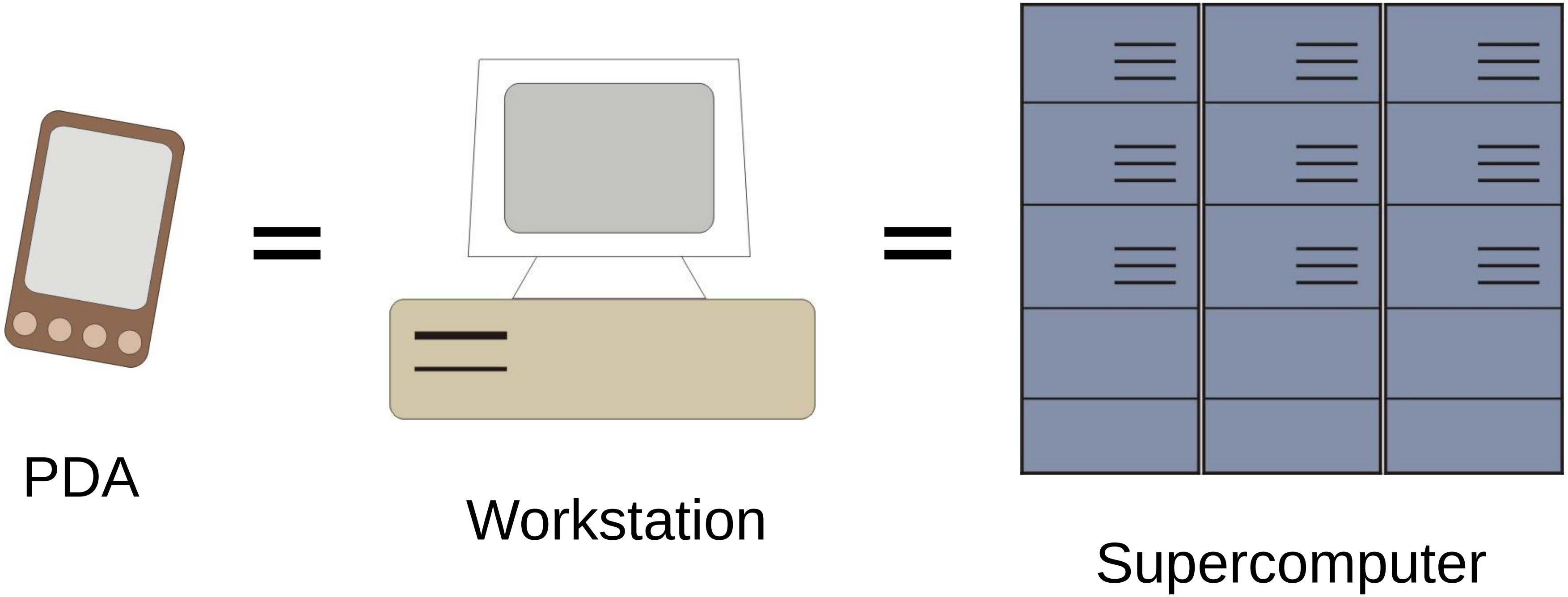
- Important to understand the components and how they work together.

Hardware vs. Software

- It’s not either/or – both are components of a computer system.
- Even if you specialize in one,
  you should understand capabilities and limitations of both.

Big Idea #1: Universal Computing Device

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

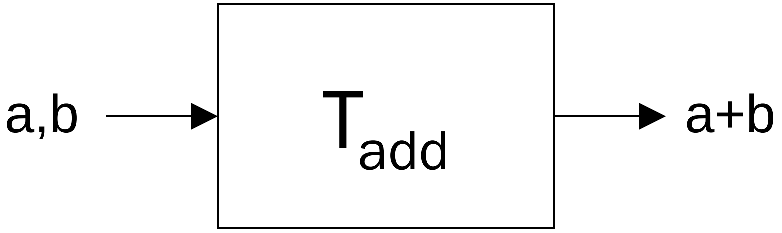


PDA = Workstation = Supercomputer

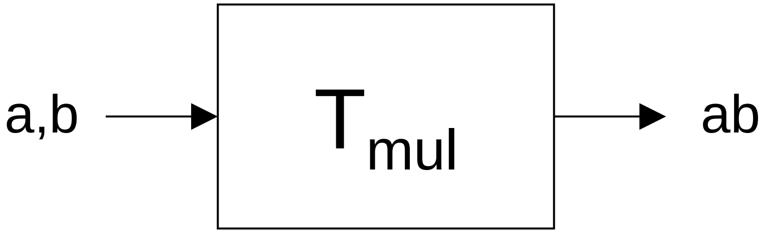
Turing Machine

Mathematical model of a device that can perform any computation – Alan Turing (1937)

Every computation can be performed by some Turing machine. ***(Turing's thesis)***



Turing machine that adds

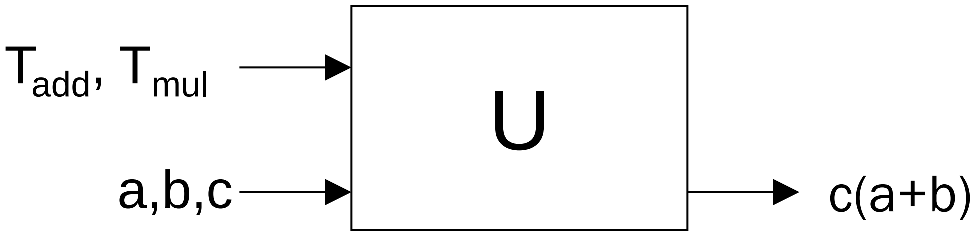


Turing machine that multiplies

Universal Turing Machine

A machine that can implement all Turing machines
-- this is also a Turing machine!

- **inputs: data, plus a description of computation (other TMs)**



Universal Turing Machine

U 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

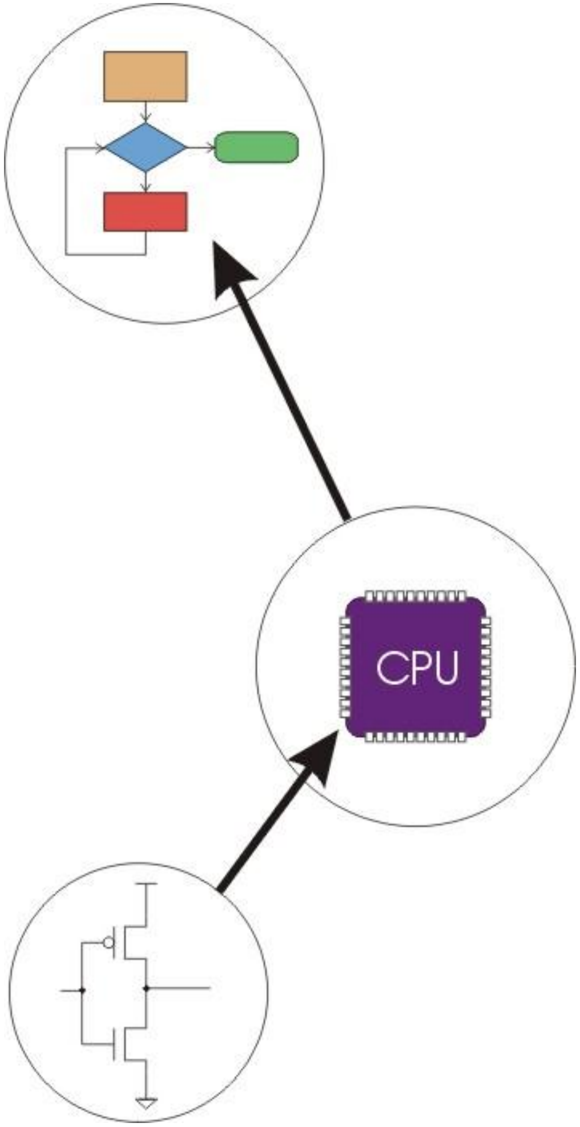
In theory, computer can *compute* anything that's possible to compute

- **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

- - - - - - - - - - - - - - - - - - - -

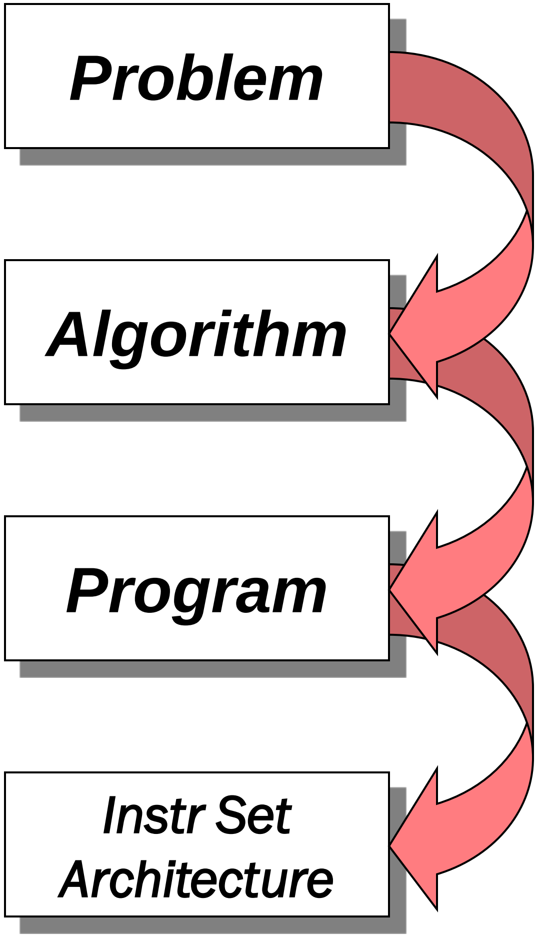
Circuits

- - - - - - - - - - - - - - - - - - - -

Devices

How do we solve a problem using a computer?

A systematic sequence of transformations between layers of abstraction.



Problem

Algorithm

Program

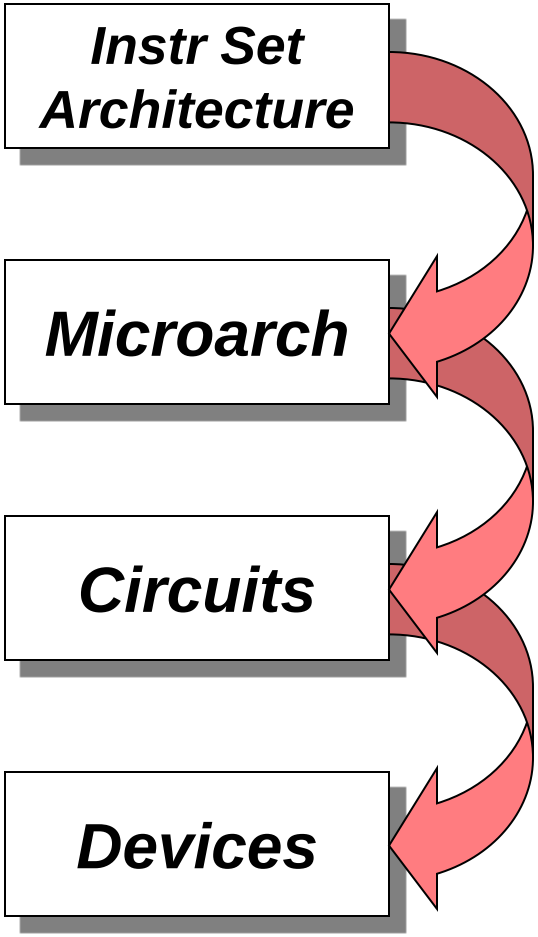
Instr Set Architecture

Software Design:
choose algorithms and data structures

Programming:
use language to express design

Compiling/Interpreting:
convert language to
machine instructions

How do we solve a problem using a computer? (cont.)



Instr Set Architecture

Microarch

Circuits

Devices

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 (Ex. Time flies like an arrow)**

Algorithm

- **step-by-step procedure, guaranteed to finish**
- **definiteness, effective computability, finiteness**

Program

- **express the algorithm using a computer language (mechanical 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

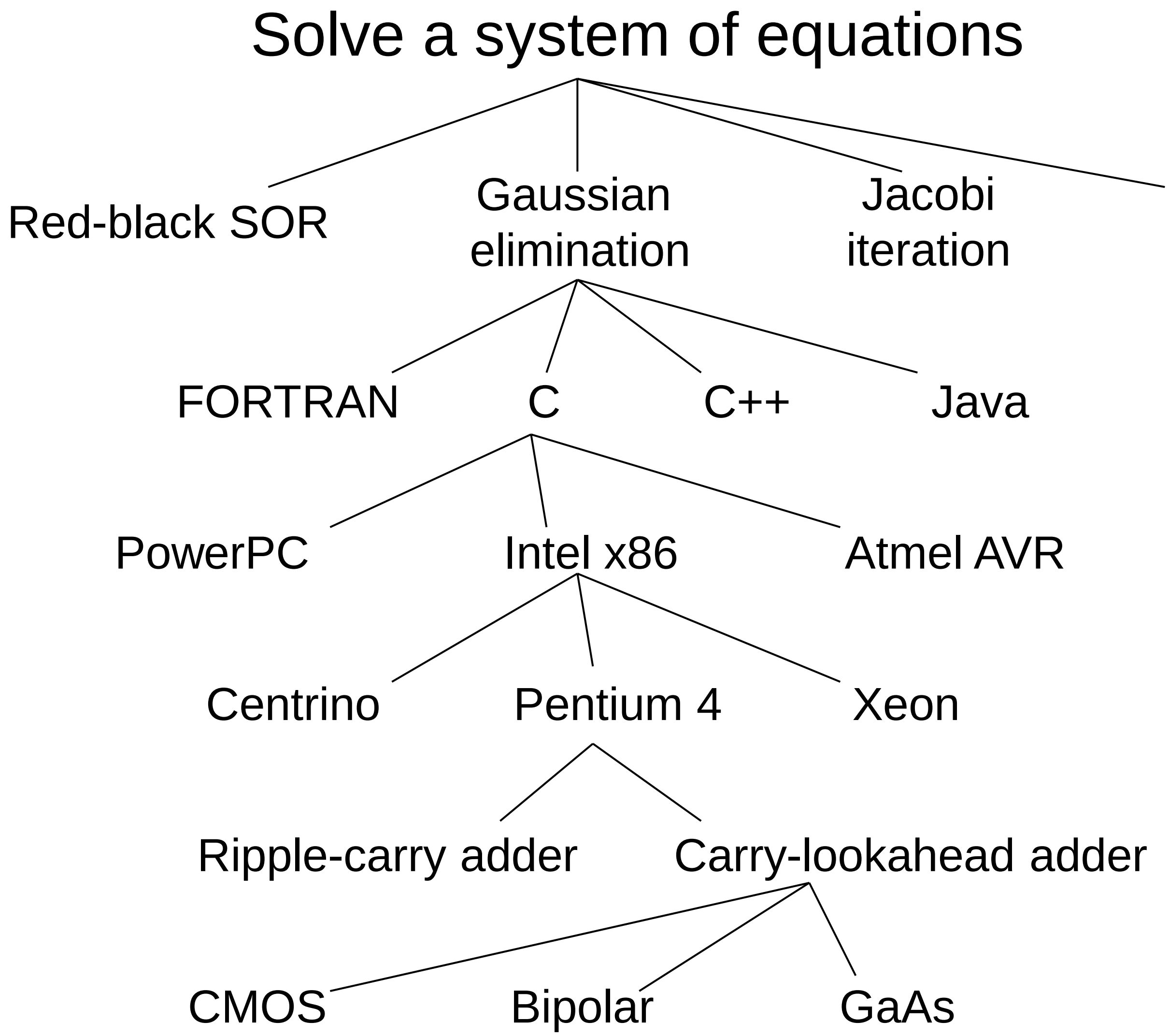
Logic Circuits

- 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



Solve a system of equations

Red-black SOR

Gaussian elimination

Jacobi iteration

Multigrid

FORTRAN

C

C++

Java

PowerPC

Intel x86

Atmel AVR

Centrino

Pentium 4

Xeon

Ripple-carry adder

Carry-lookahead adder

CMOS

Bipolar

GaAs

Tradeoffs:

cost

performance

power

(etc.)

Course 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?