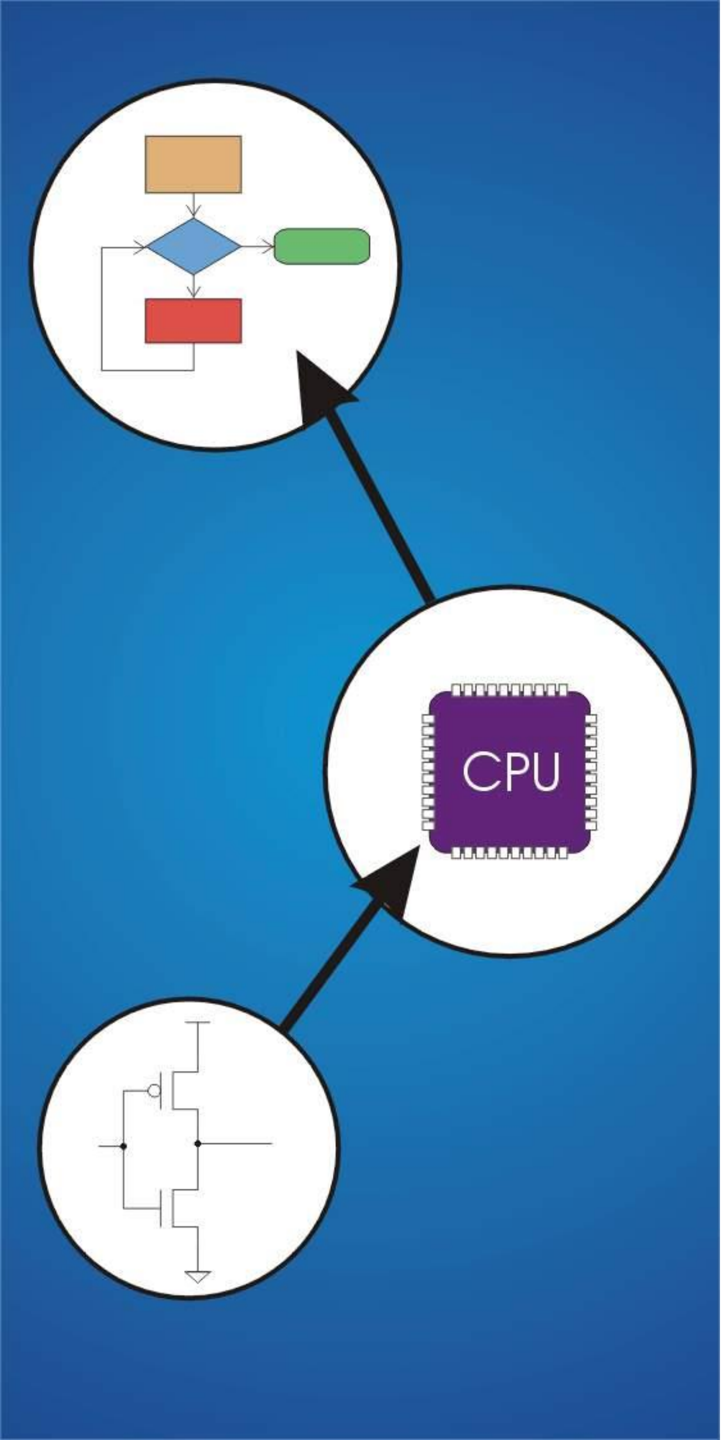
Introduction to Computer Engineering

CS/ECE 252, Fall 2014

Prof. Guri Sohi

Computer Sciences Department

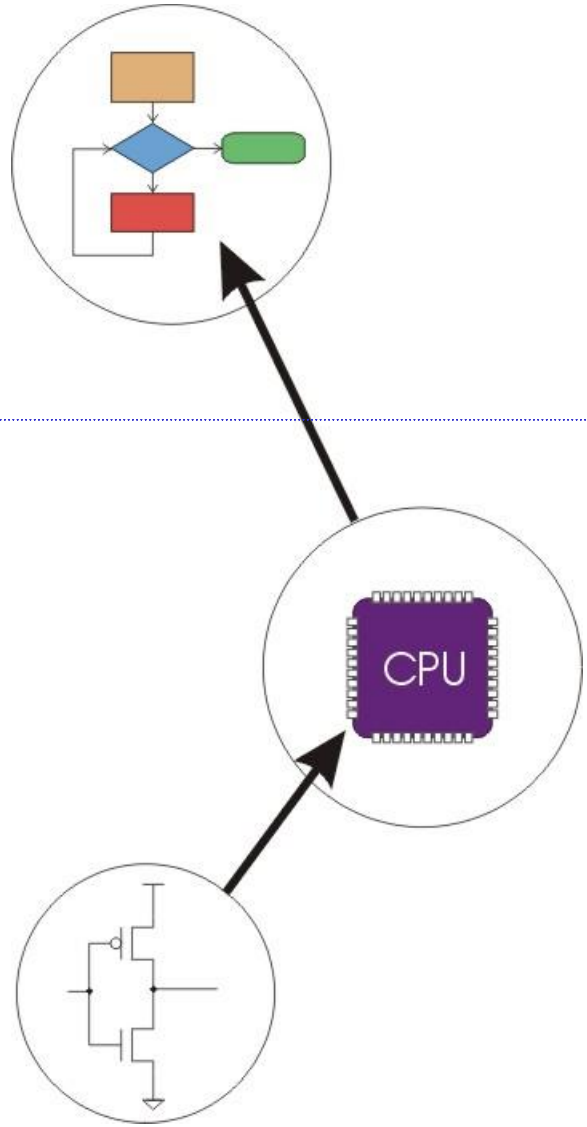
University of Wisconsin – Madison



Chapter 1
Welcome Aboard

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

Computer System: Layers of Abstraction



Application Program

Algorithms

Language

Software

Hardware

Instruction Set Architecture
(and I/O Interfaces)

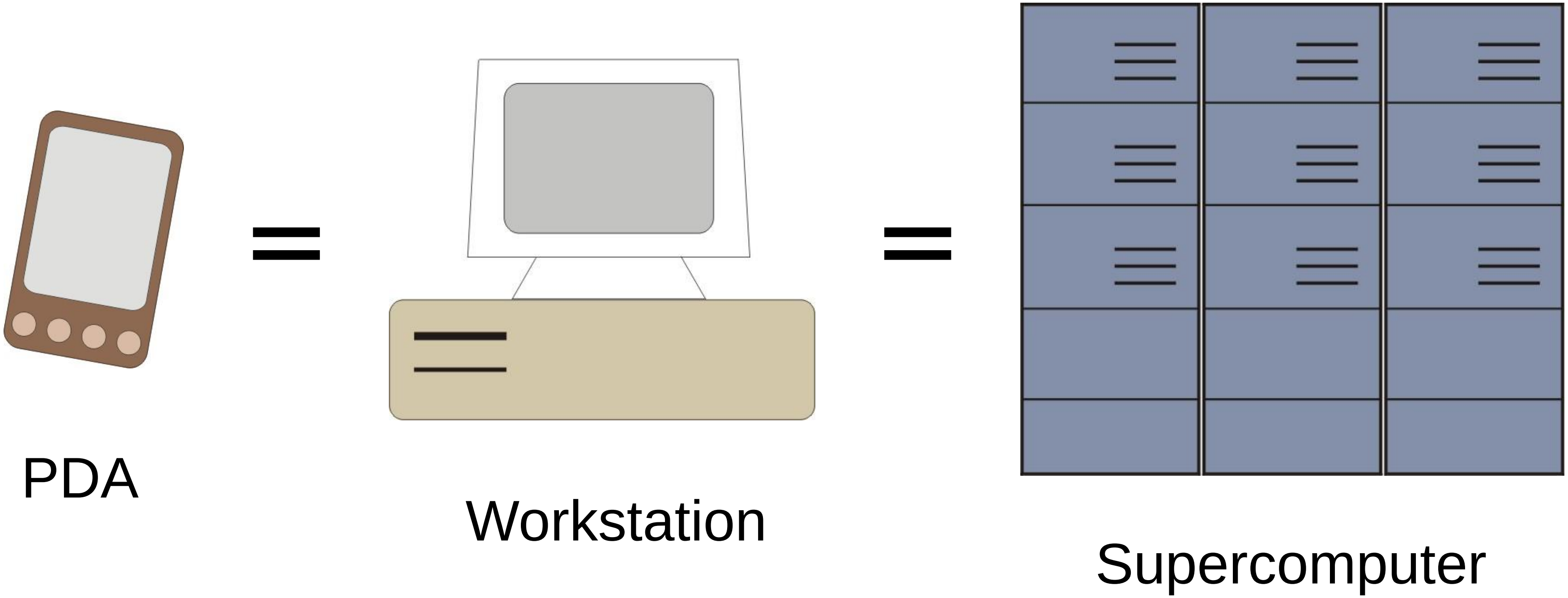
Microarchitecture

Circuits

Devices

Big Idea #1: Universal Computing Device

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

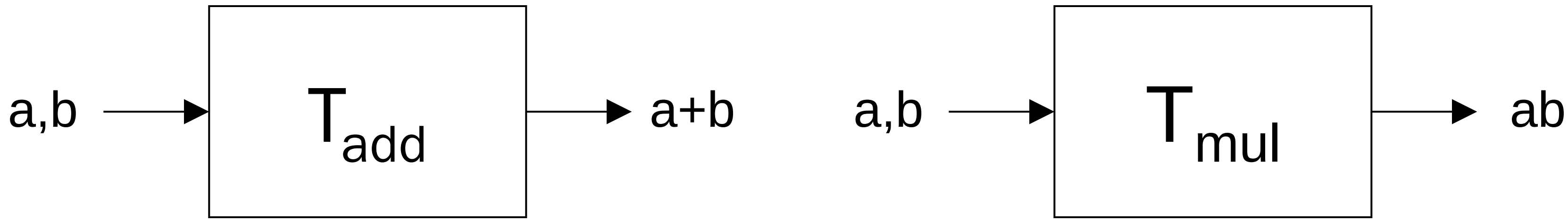


Turing Machine

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)***



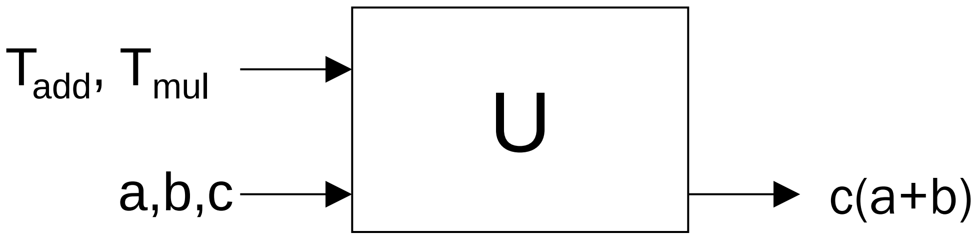
Turing machine that adds

Turing machine that multiplies

Universal Turing Machine

Turing described a Turing machine that could implement all other Turing machines.

- **inputs: data, plus a description of computation (Turing machine)**



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, and vice versa**

Therefore, 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***

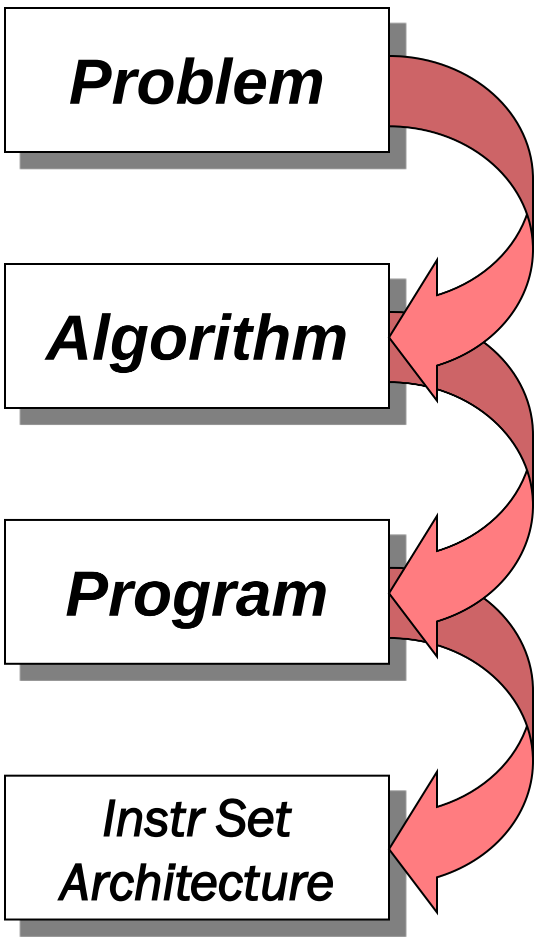
In practice, *solving problems* involves computing under constraints.

- **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

How do we solve a problem using a computer?

A systematic sequence of transformations between layers of abstraction.

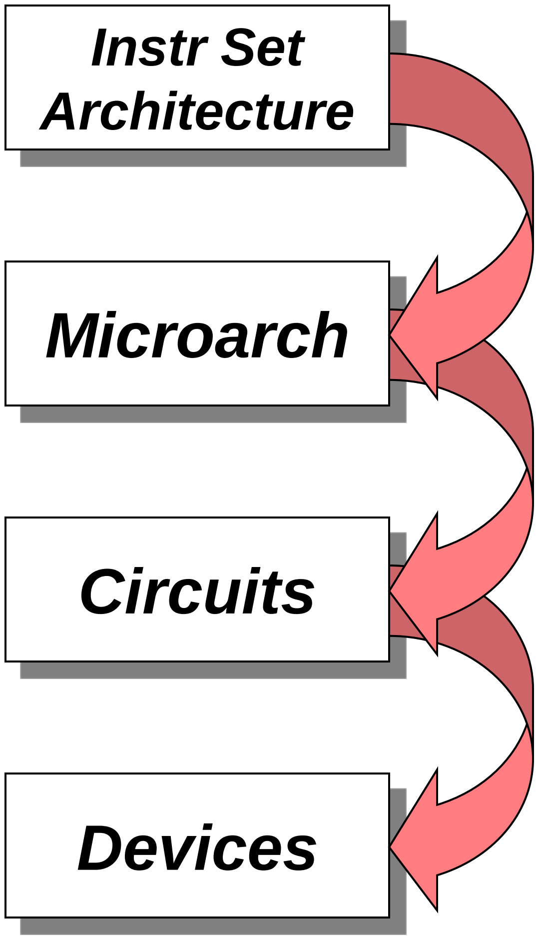


Software Design:
choose algorithms and data structures

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**

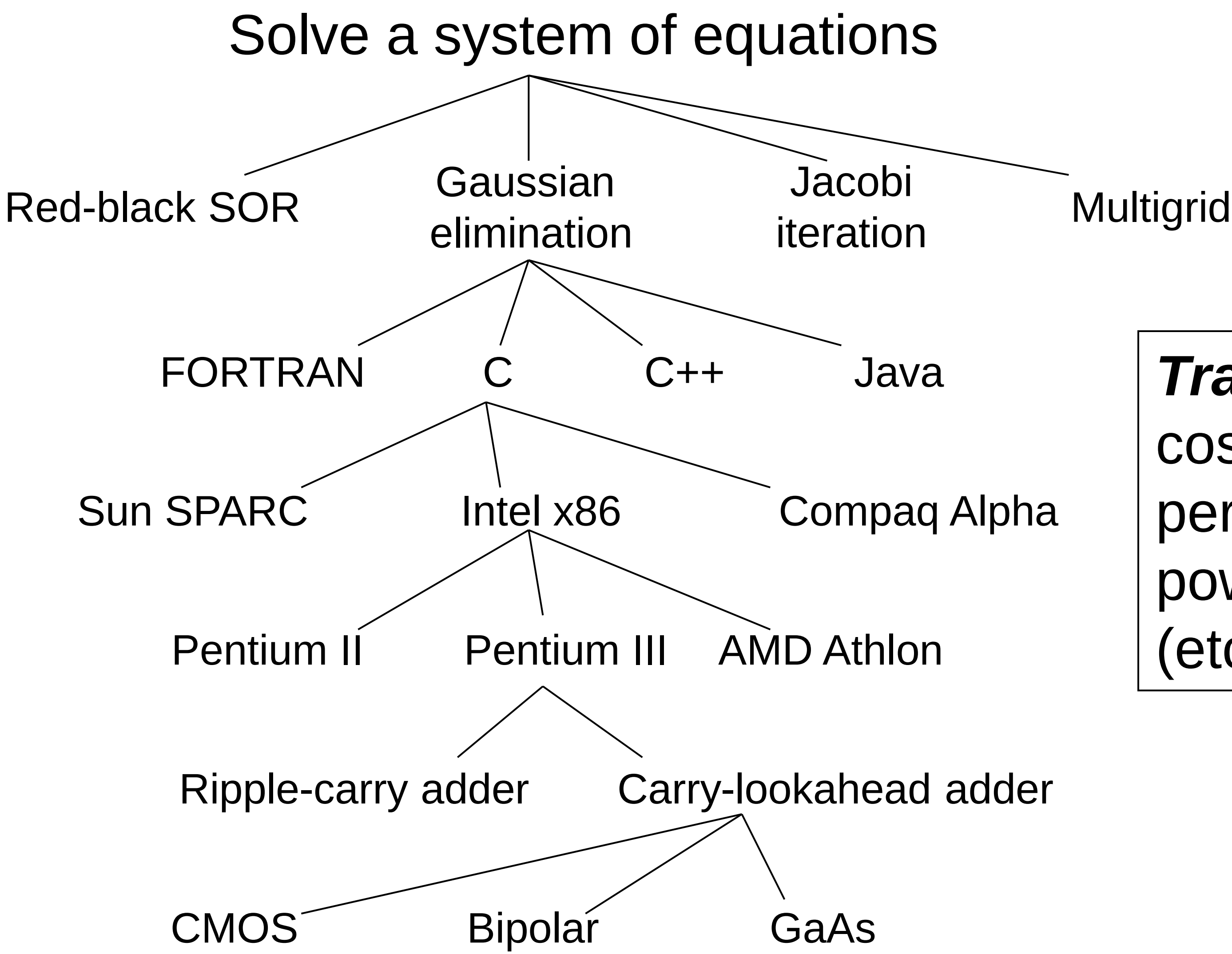
Logic Circuits

- **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.)

What's Next

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?