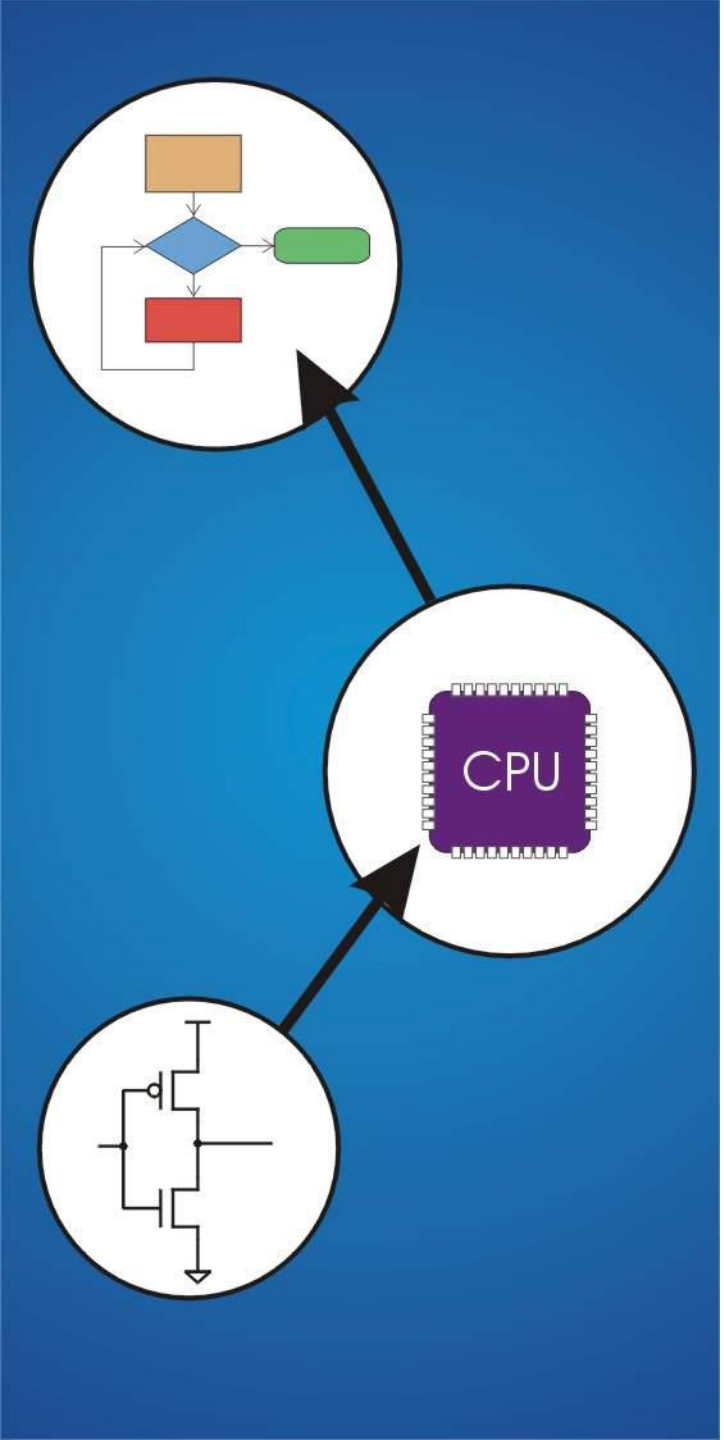


CS270

Computer Organization

Fall 2017

Lecture Goals

Review course logistics

- Assignments & quizzes
- 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, or directly**

Assignments & Quizzes

Assignments are posted on the course website:

Weekly assignments (mostly):

- **Programming (C, LC-3), Logisim**
- **submission mode and deadline varies.**

Programming assignments are submitted in electronic form 10PM.

- **There may be a "Late period", perhaps 1 hour 59 min, with 20% off. Sometimes there may not be a late priod.**
- **Regrading requests: through Piazza (see syllabus for policies).**

Quizzes: can be on-line (canvas) or in-class (using iCliker)

Policies

Grading Criteria

- Assignments (30%)
- Recitations (10%)
- Quizzes(10%) online and in-class (iClicker)
- Two Midterm Exams (15% each)
- Final Exam (20%)

Late Policy

- On-time = full points, late submission= 20% penalty within specified late period (if any)

Academic Integrity

- http://www.cs.colostate.edu/~info/student-info.html
- Do your own work
- Cannot copy and paste *any* code, unless provided by us

People

Instructors:

- Section 1: Dave Matthews
- Section 2: Yashwant K. Malaiya

Graduate Teaching assistants:

- GTA 1: Fahad Ullah
- GTA 2: Matthew Dragan

Undergraduate Teaching Assistants:

- UTA 1:, Jason Stock
- UTA 2:, Sean Thunquest
- UTA 3:, Alex Madey

Office hours/locations

- See course website

About me

Instructor: …

Background: …

Office hours/locations

- **…**

Organization

Order used in the class: Top down

1/3 C programming: data types, language syntax, variables and operators, control structures, functions, pointers and arrays, memory model, recursion, I/O, data structures

1/3 Instruction set architecture: machine/assembly code, instruction formats, branching and control, LC-3 programming, subroutines, memory model (stack)

1/3 computer hardware: numbers and bits, transistors, gates, digital logic, state machines, von Neumann model, instruction sets, LC-3 architecture

Top Down Perspective

- **Multilayered view:**
  - **Higher layers serves as the specification.**
  - **Lower layer implements provides the implementation**
- **We will see**
  - **How a higher level language (C) is implemented by a processor instruction-set architecture (ISA), LC-3 in our case ?**
  - **How an ISA is implemented using digital circuits?**
  - **How are digital circuits implemented using transistors?**
  - **And so on …**

Grading Criteria

Letter Grade	Points
A	≥90%
B	≥80%
C	≥70%
D	≥60%

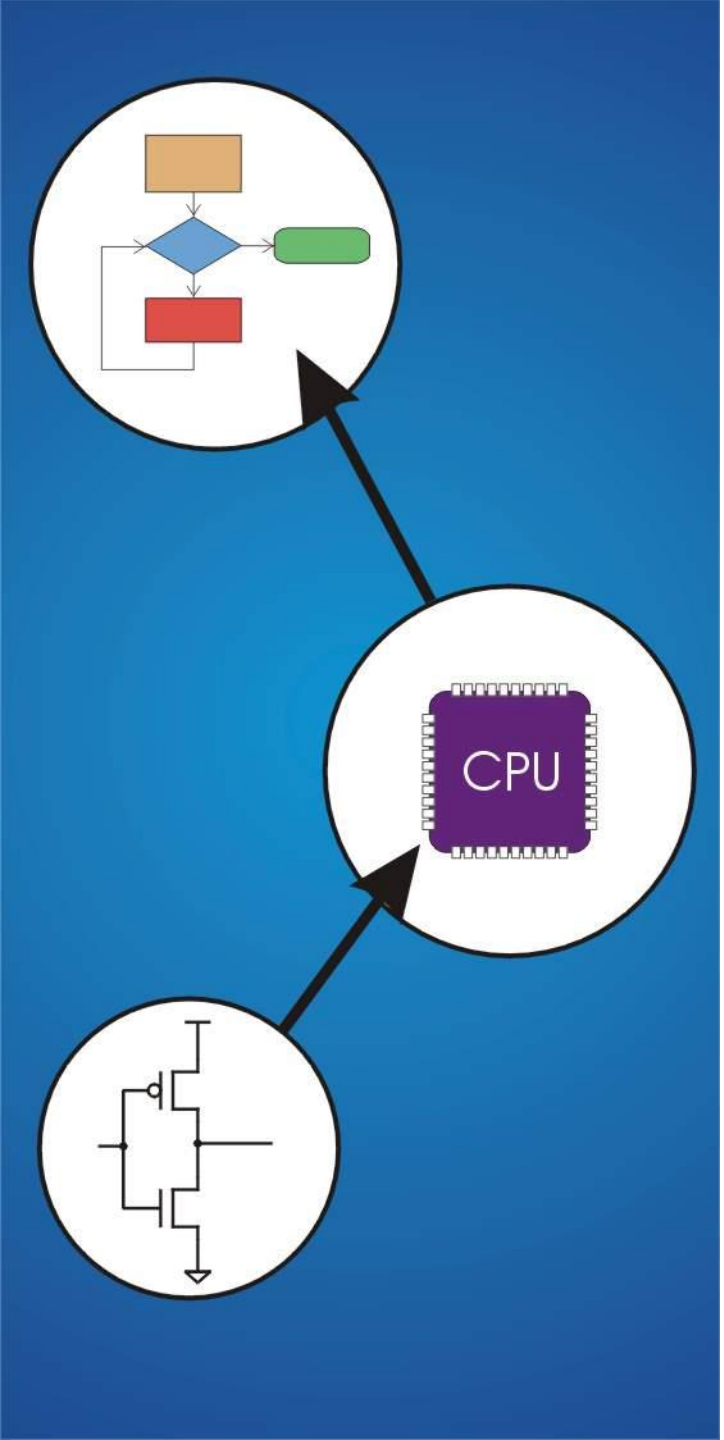
- **We will not cut higher than this, but we may cut lower.**
- **Your average score on exams must be ≥60% to receive a passing grade in this course.**

How to be successful in this class

1) **Attend all classes and recitations.**
2) **Take the in-class and on-line quizzes as required.**
3) **Do all the assignments,**
   1) **ask questions (early! (but not *too* early!)) if you run into trouble.**
4) **Take advantage of lab sessions where help is available from instructors.**
5) **Read the textbook.**

Text book:
Introduction to Computing Systems:
From Bits and Gates to C and Beyond
2nd Edition

Yale N. Patt and Sanjay J. Patel



Chapter 1
Welcome Aboard

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?

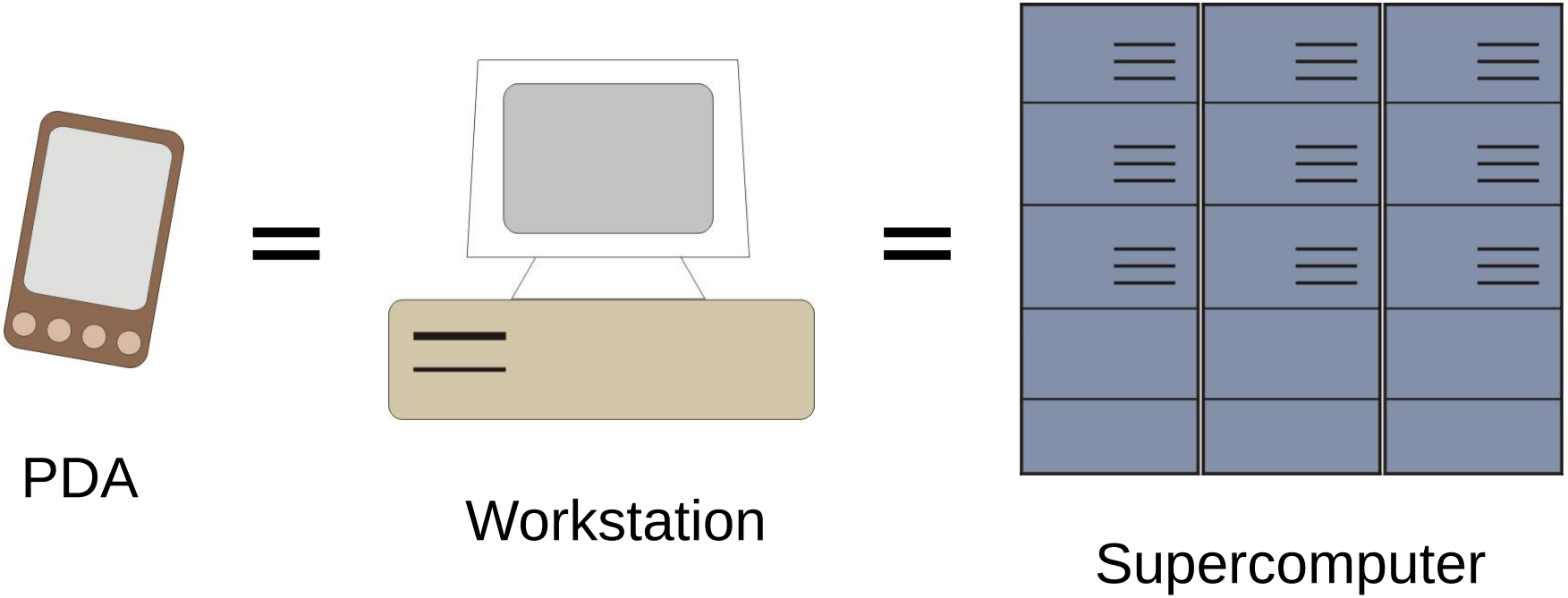
- **Important to understand the components and how they work together.**

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.

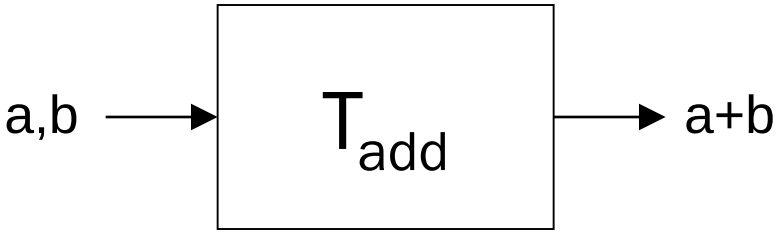


Turing Machine

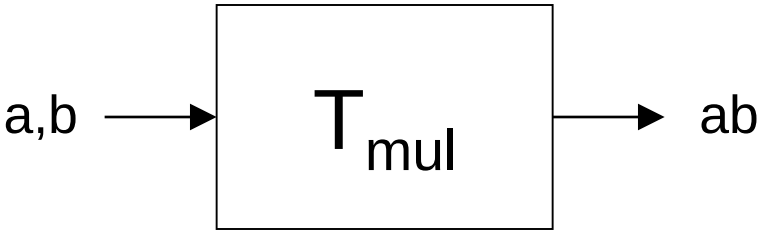
Mathematical model of a device that can perform any computation – Alan Turing (1937)

- **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

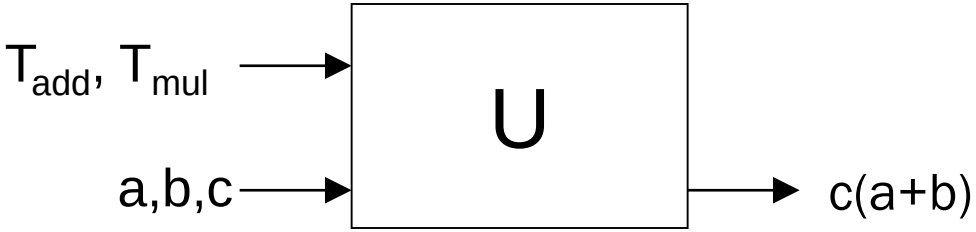
For more info about Turing machines, see http://www.wikipedia.org/wiki/Turing_machine/

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, plus a description of computation (other TMs)**



Universal Turing Machine

U is <u>programmable</u> – 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

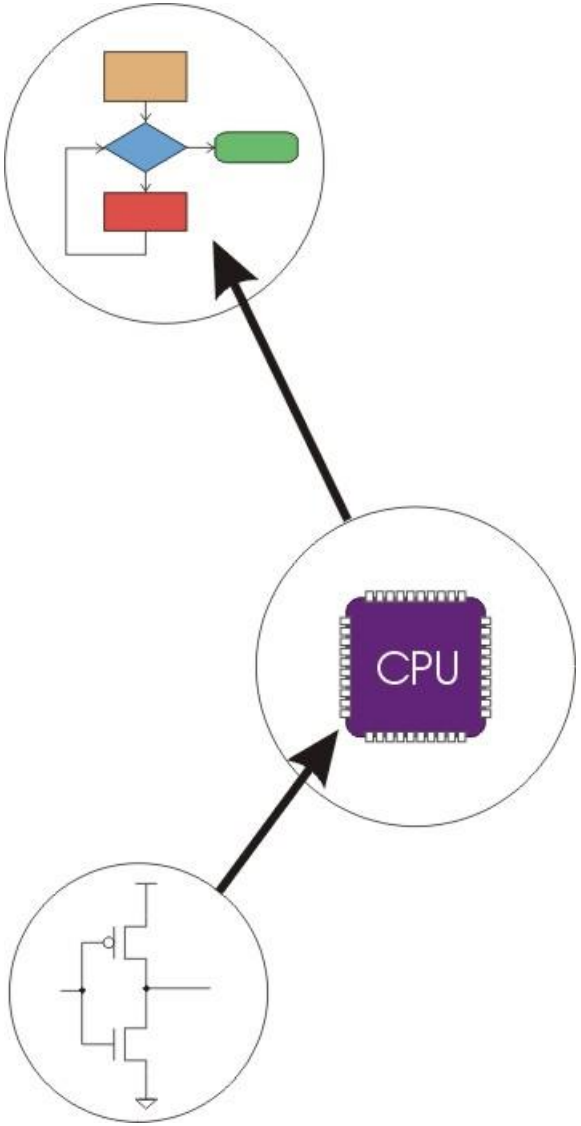
In theory, computer can *compute* **anything that's possible to compute**

- **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

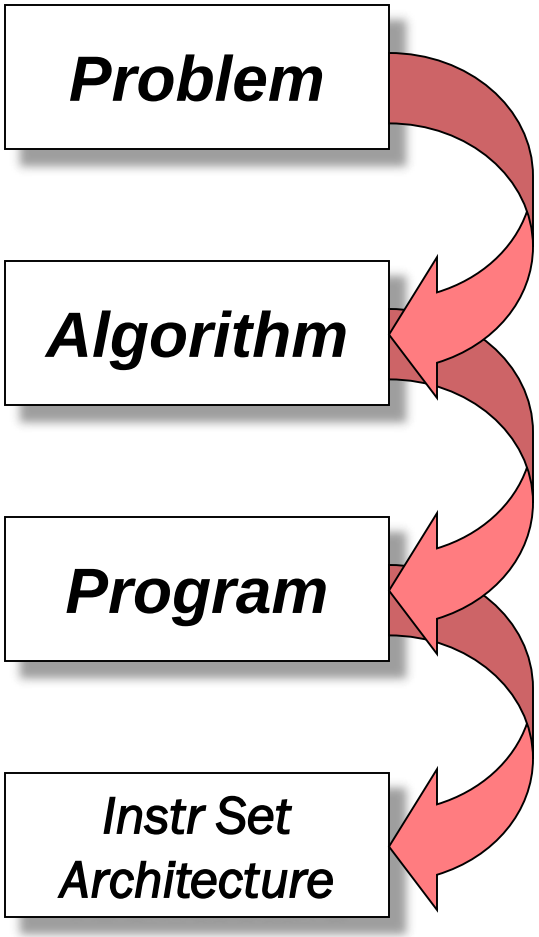
Microarchitecture

Digital Circuits

Devices

How do we solve a problem using a computer?

A systematic sequence of transformations between layers of abstraction.

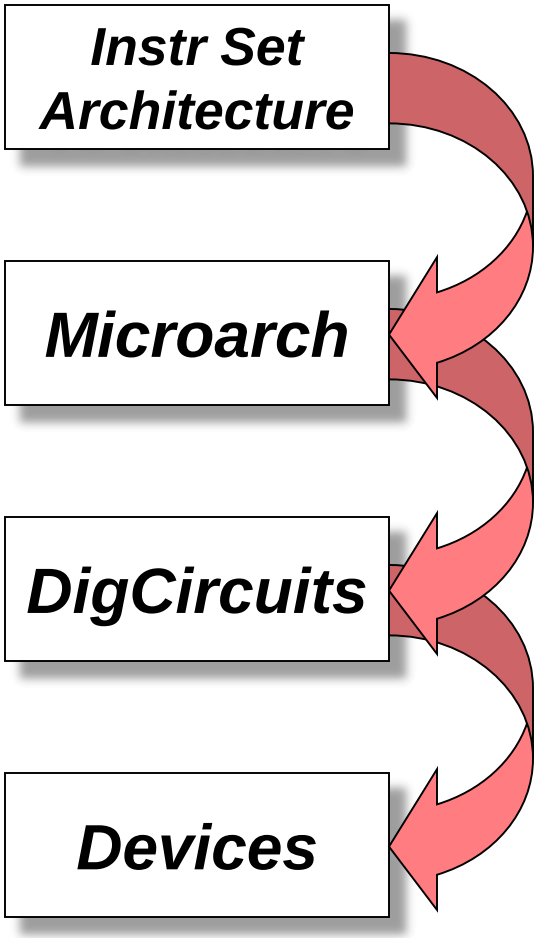


Software Design:
choose algorithms and data structures

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 processor (CPU) 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

iClicker Quiz (trial)

Registration

- **Please register your iClicker using canvas and bring it every time**
- **Ensure you are using the right channel**

Quiz: Pick one: Instruction Set Architecture (ISA)

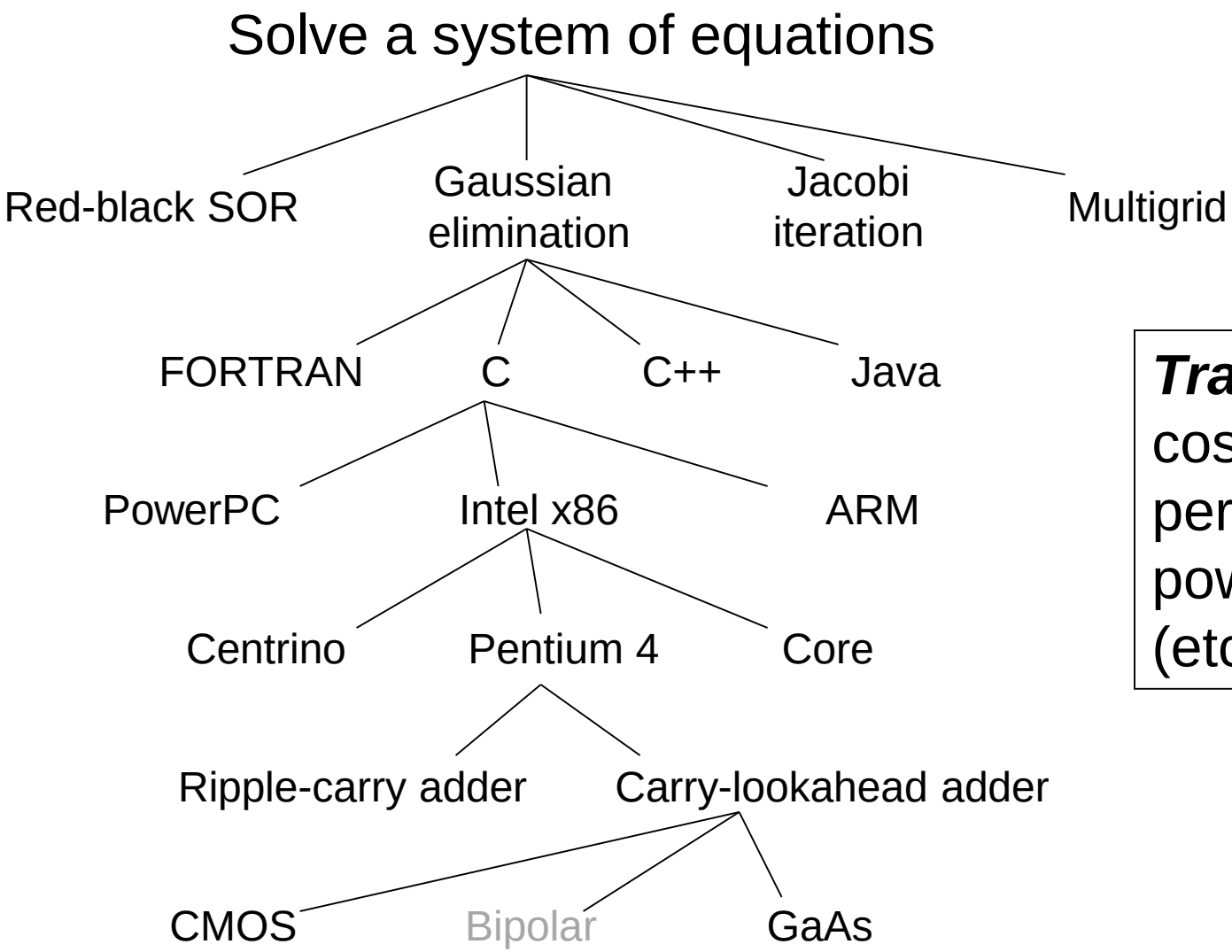
- **A. specifies the set of instructions the CPU can perform,**
- **B. Architecture of a high level language**
- **C. How transistors are used to form digital circuits**
- **D. Architecture of a C program**
- **E. All of the above**

iClicker Quiz (trial) Answer

Quiz: Pick one: Instruction Set Architecture (ISA)

- **A. specifies the set of instructions the CPU can perform**
- **B. Architecture of a high level language**
- **C. How transistors are used to form digital circuits**
- **D. Architecture of a C program**
- **E. All of the above**

Many Choices at Each Level



Tradeoffs:
cost
performance
power
(etc.)

Course Outline

Bits and Bytes

- How do we represent information using electrical signals?

C Programming

- How do we write programs in C?
- How do we implement high-level programming constructs?

Instruction set architecture/Assembly language

- What operations (instructions) will we implement?
- 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?

Digital Logic and processor architecture

- How do we build circuits to process and store information?
- How do we build a processor out of logic elements?

Computer systems: what is next?