# COMP 2280 - Introduction to Computer Systems

## Module 1 - Introduction

# Introduction

- Learning objectives for this course:
  - Understand how computers really work.
  - Understand how programs are executed on a computer.
  - Learn how to program a computer using assembly language.
- Reasons for learning these topics:
  - Be able to write better programs.
  - Make better software/hardware decisions.
  - Understanding relationships between various types of software.
- This is an introductory course for studying the organization/architecture of a computer system.
  - COMP 3370 will cover more advanced topics.

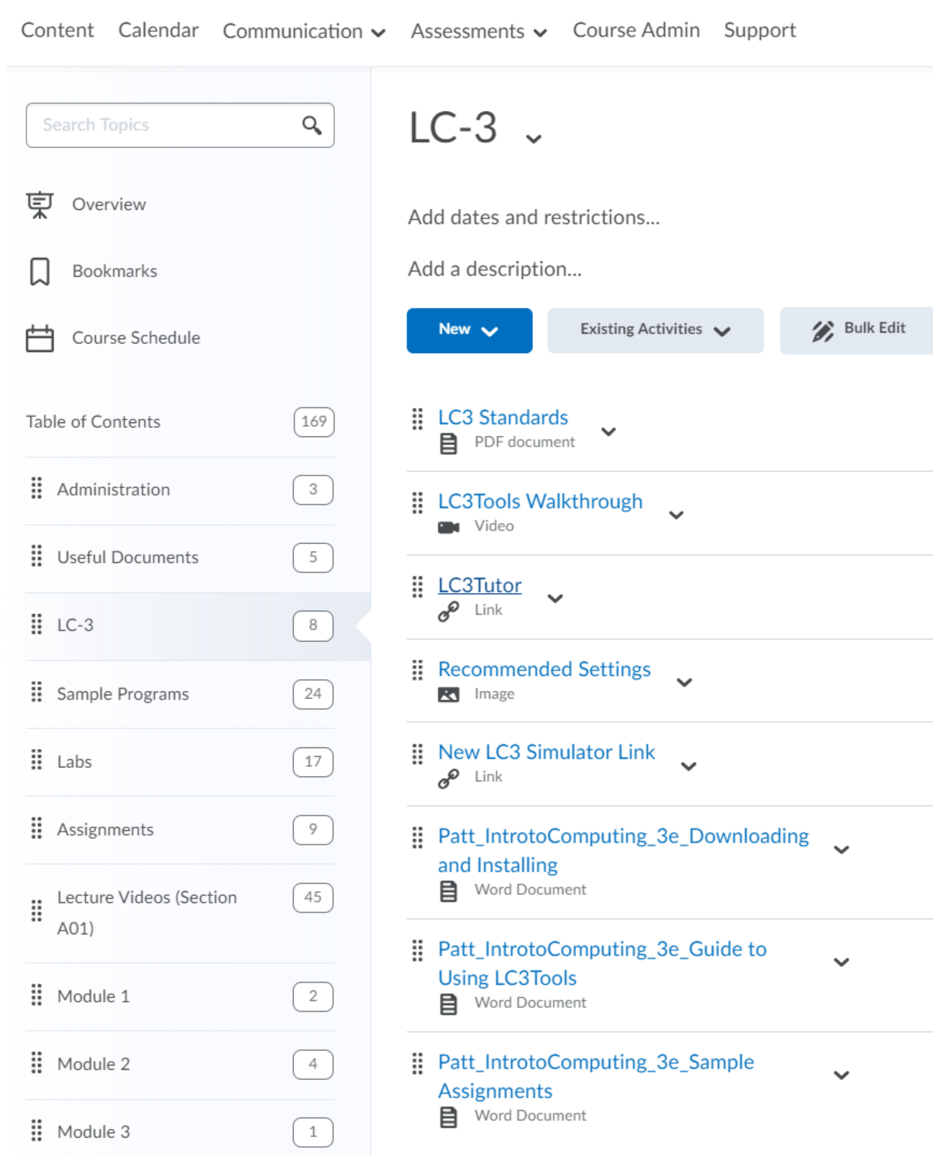
# Get the book

- Look into **buying/borrowing/acquiring** the course book
  - **"Introduction to Computing Systems: from Bits & Gates to C/C++ & Beyond, third edition, by Patt and Patel, McGraw Hill."**

- Here are some links for purchase, the 3rd edition came out recently (Sept. 3 2019).
  - The 2nd edition *might* suffice but there are always small differences between them so be wary.
    - DIGITAL - https://www.vitalsource.com/en-ca/products/ise-introduction-to-computing-systems-from-bits-amp-yale-patt-v9781260569667
    - https://www.amazon.ca/Introduction-Computing-Systems-Gates-Beyond-ebook/dp/B07VWKMJBX/ref=sr_1_2
    - https://www.mheducation.com/highered/product/introduction-computing-systems-bits-gates-c-c-beyond-patt-patel/M9781260150537.html
    - https://www.mheducation.ca/ise-intro-computing-systems-bits-gates-c-beyond-9781260565911-can-group

# Get LC3

- Install LC3 early in case of issues
- ***HIGHLY*** recommended you follow the in-class examples and play around with the sample programs given.
- LC3Tutor is also a great resource
- http://lc3tutor.org/



# Grades

- 5 Labs - 10%
- 4 Assignments - 20%
- Midterm Test - 30% (tentative date: sometime in the middle two weeks of July)
- Final Exam - 40%

- The minimum percentage grade required for A+, A, B+, B, C+, C, D will be 90%, 80%, 75%, 70%, 65%, 60%, 50% respectively, ***plus or minus 5%.***
- Any adjustment will be made to ensure grades are assigned fairly.

| Grade | Range |
|---|---|
| A+ | 90% – 100% |
| A | 80% – 89% |
| B+ | 75% – 79% |
| B | 70 – 74% |
| C+ | 65% – 69% |
| C | 60% – 64% |
| D | 50% – 59% |
| F | < 50% |

# Two Recurring Themes

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

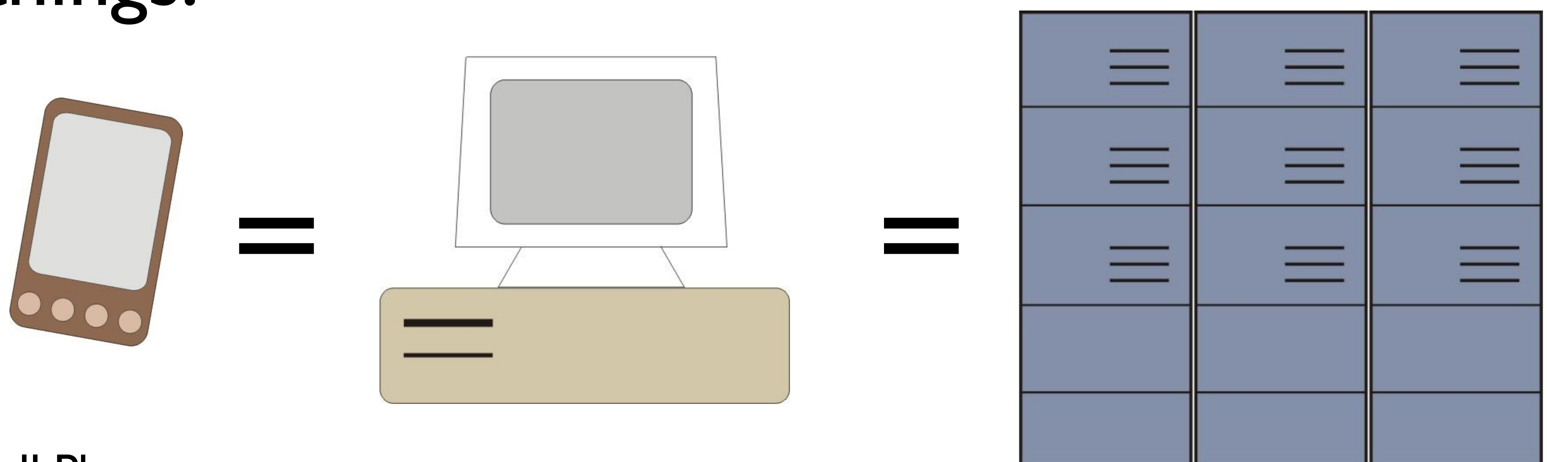
    Can drive a car without knowing how
    the internal combustion engine works.
  - …until something goes wrong!

    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.



Cell Phone

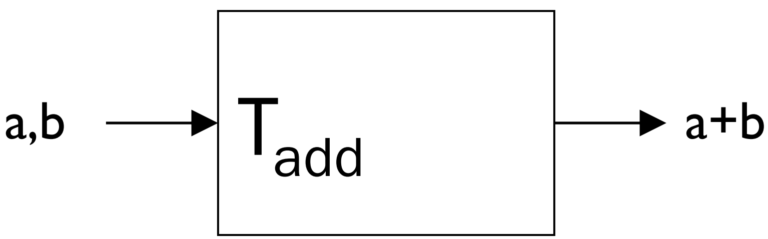
Workstation

Supercomputer

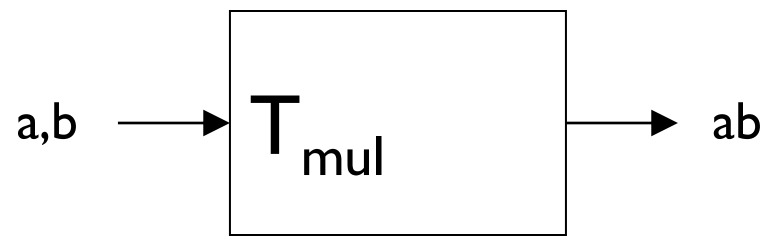
# Turing Machine

- 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*



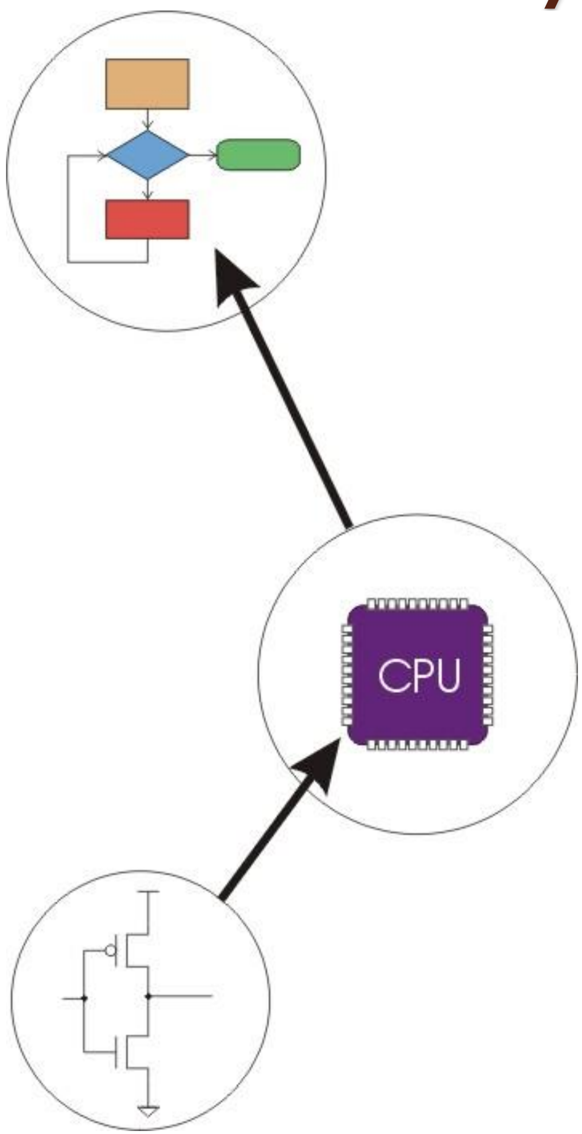

*Turing machine that multiplies*

# 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**

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

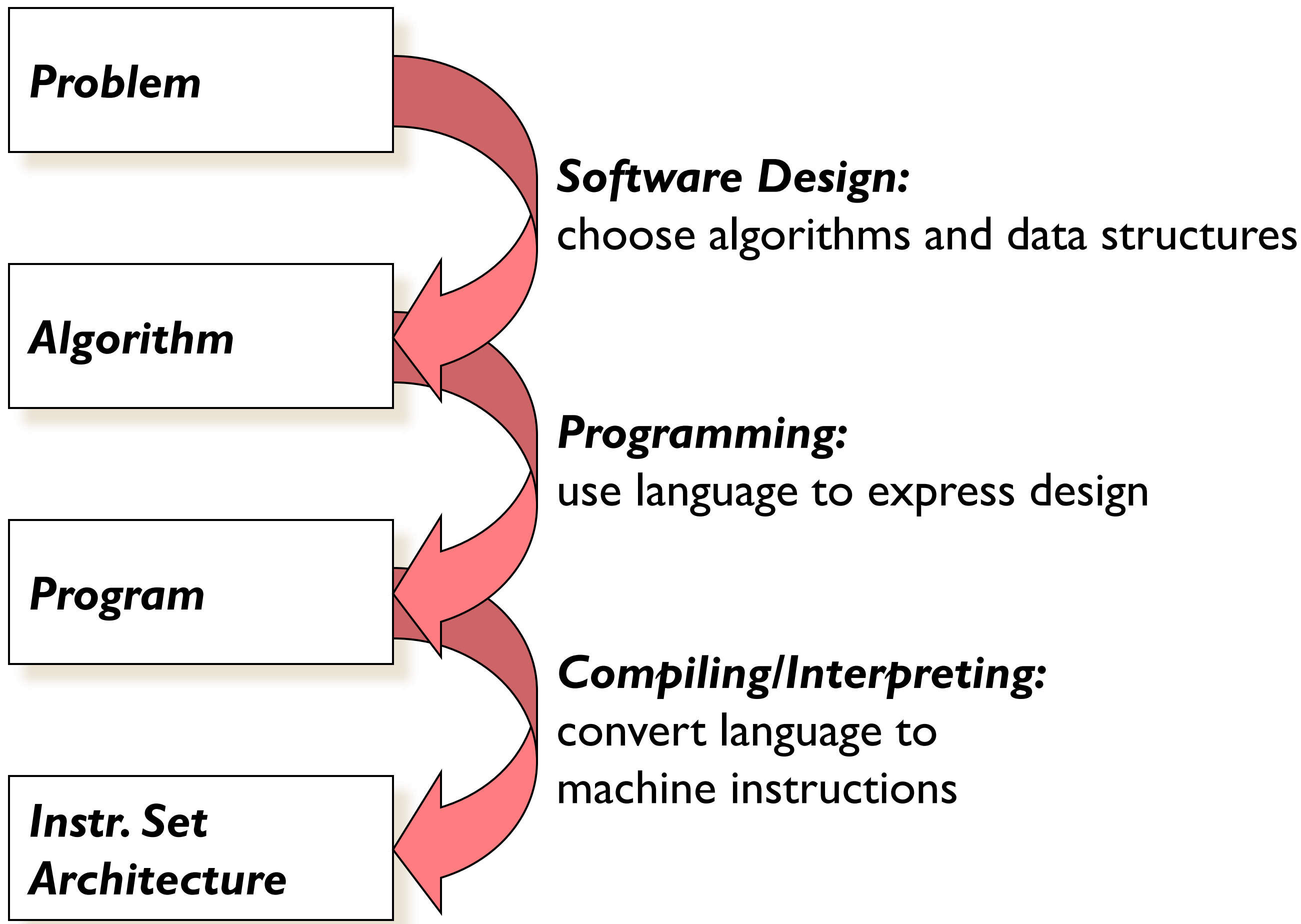
**Circuits**

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

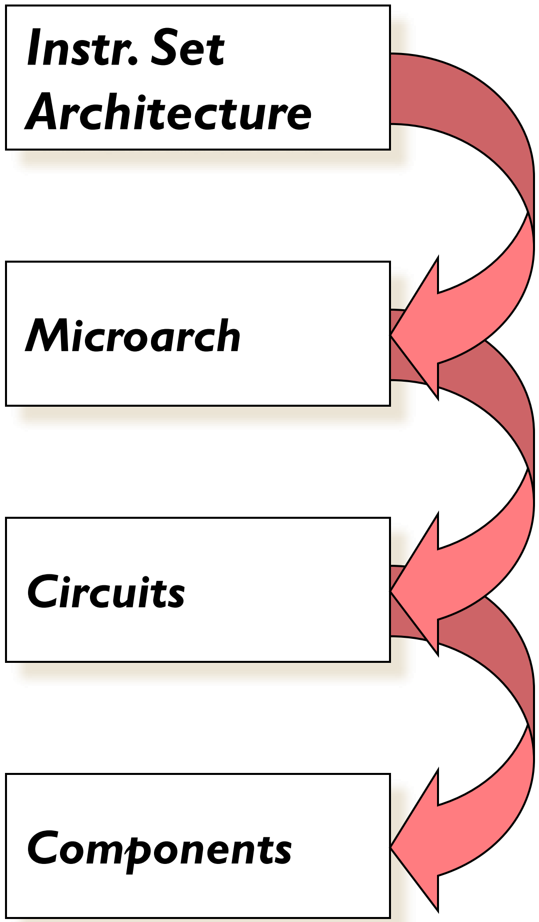
**Components**

# How do we solve a problem using a computer?

- A systematic sequence of transformations between layers of abstraction.



# 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 modules

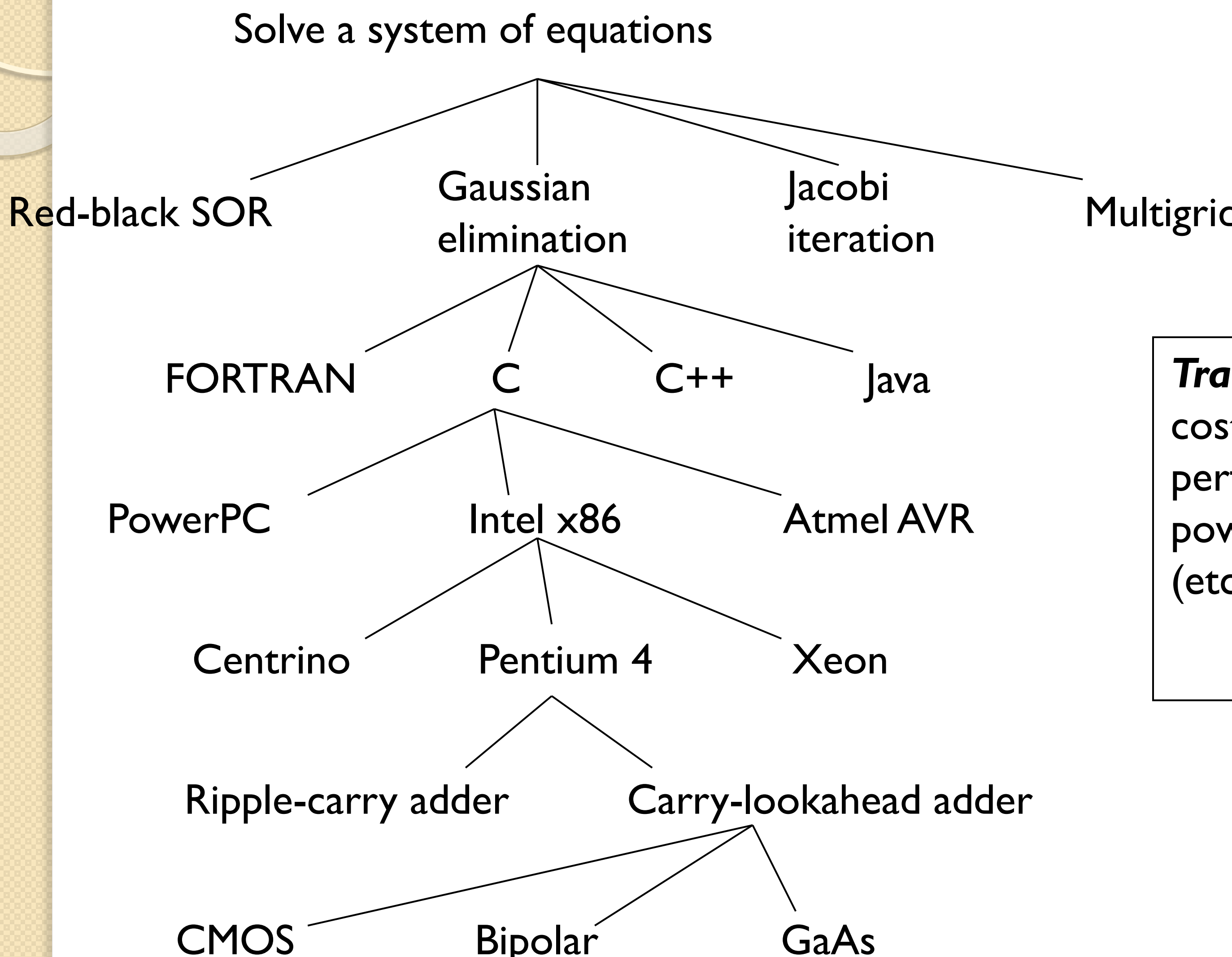
# Descriptions of Each Level

- 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



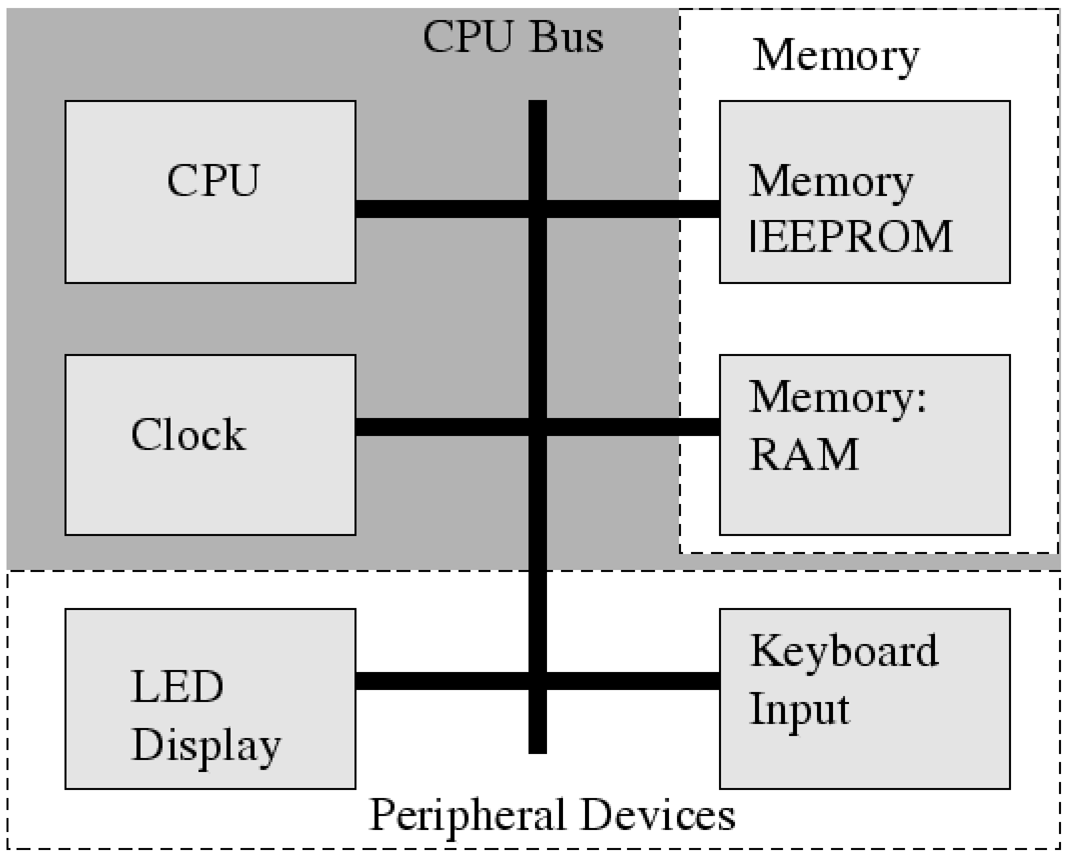

***Tradeoffs:***
cost
performance
power
(etc.)

# Basic Components

- **Von Neumann Machines** consists of
  - CPU, Main memory, I/O
  - stored-program computer
  - executes instructions sequentially

# Basic Components

- Here is a generic (Von Neumann) computer architecture.



# Major Components

- **Arithmetic Logic Unit** (ALU)
  - Performs the desired operations on the data.
  - Really several "functional units" in modern machines.
- **Control Unit**
  - Coordinates the activities of the computer system and its components (including ALU).
- *Note:* ALU and CU make up the CPU

# Major Components

- **Input Unit**
  - accepts *coded* information from human operators via hardware devices (e.g. keyboards, mice, etc.), or from other computers via some digital communication medium (Ethernet or …).
- **Output Unit**
  - Results are sent to the outside world (users) via one or more output devices (printers, CRTs, …).
- *Note*: Input and output units are usually grouped together as the I/O unit.

# Major Components

- **Memory Unit**
  - Stores programs and data for immediate access
  - Divided into two classes: Primary *memory* and secondary *storage*
  - Information to be processed by the central processing unit is stored in primary memory (a.k.a. main memory)

# Instructions & Data

- Every CPU has an instruction set architecture (ISA) which can be used to write programs.
- Instructions specifies the arithmetic and logic operations to be performed.
- Programs are stored in memory for execution.
- The CPU fetches the program's instructions from memory one at a time and executes the specified operations.
- Programs usually have to work with data.
- Data can be numbers or other *encoded* data.
- Data must be encoded in a suitable format as required by the machine.

# A Sample Instruction

- A typical instruction in the LC-3 machine is
  - ADD R1, R2, R3
- In memory, this instruction is encoded using16 bits as
  - 0001 001 010 0 00 011 (spaces are for readability only)
- This instruction adds the value in register R2 of the CPU to the value in a register R3 of the CPU and stores the result in register R1 of the CPU.
- The steps involved in executing this instruction are:
  - Transfer the instruction from main memory into the processor (fetching & decoding the instruction)
  - fetch the operands for access by the ALU
  - perform the addition
  - store the result back into R1
- The steps above is known as an *instruction cycle*.

# Next Steps

- A large portion of this course is dedicated to programming the LC-3 machine.
- To do this, we need to know the fundamentals.
  - Learn how data is represented (so we can manipulate it).
  - Learn how to write programs in assembly language.
  - Learn standard programming constructs and write them using assembly language.
- Then we'll come back to all of this stuff to figure out how it works and its relationships to the major components of the machine.

# Next Steps

- Next lecture:
  - Learn how data is encoded.
  - We will consider encodings for the following data types:
    - Signed and unsigned integers,
    - Floating point numbers,
    - Characters.

# Recap

- 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?
- I/O, Traps, and Interrupts
  - How does processor communicate with outside world?
- Assembly Language Programming
  - How do we use processor instructions to implement algorithms?
  - How do we write modular, reusable code? (subroutines)