What is a computer?

- What is computation?
- There are many sorts of computing devices, they fall into two categories:
  - Analog: machines that produce an answer that measures some continuous physical property such as distance, light intensity, or voltage. Examples:
  - Digital: machines that perform computations by manipulating a fixed finite set of elements. Examples:
- The difficulty with analog devices is that it is very hard to increase their accuracy.
- Before modern digital computers, the most common digital machines were adding machines.
- Adding machines perform exactly one operation.
- Modern general-purpose digital computers also perform one operation, but their operation is to accept a set of instructions that tell it how to do any sort of computation.

How do we get the electrons to do the work?

- We describe our problems in English or some other natural language. Computer problems are solved by electrons flowing around inside the computer.
- It is necessary to transform our problem from a natural language to the voltages that influence the flow of electrons.
- This transformation is really a sequence of systematic transformations, developed and improved over the last 50 years, which combine to give the computer the ability to carry out what may appear to be very complicated tasks. In reality, these tasks must be simple and straightforward.

Levels of abstraction in digital computation

<table>
<thead>
<tr>
<th>Level</th>
<th>Discipline</th>
<th>Course/Textbook</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software level</td>
<td>Computer Science</td>
<td>CS 24x, 4xx</td>
</tr>
<tr>
<td>Algorithms &amp; Language</td>
<td>Computer Engineering</td>
<td>CEG 320</td>
</tr>
<tr>
<td>Hardware level</td>
<td>Computer Engineering</td>
<td></td>
</tr>
<tr>
<td>ISA &amp; Microarchitecture</td>
<td>Computer/Elect. Engineering</td>
<td>CEG 260/360</td>
</tr>
<tr>
<td>Logic level</td>
<td>Computer/Elect. Engineering</td>
<td></td>
</tr>
<tr>
<td>Circuits &amp; Devices</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Black Boxes

- Every level of this hierarchy that you don’t understand is a black box
- It might as well be magic.
- “Debugging by superstition”
- You are already familiar with the high-level language-level of abstraction and above.
- The objective of this course is to move the black-box boundary down to Microarchitecture!
- ISA: Essentially, the low-level tasks that a particular computer can perform. A sequence of such tasks may perform a high-level task.
- Microarchitecture: Implementation of the ISA
- This course will largely treat circuits & devices as “magical” black boxes (refer to future coursework)!
The instruction set

- High-Level Language - C, C++, or Java
  - \( A = B + C \)
- ISA Level
  - Memory-Transfer Equivalent
    - \( \text{Mem}[A] = \text{Mem}[B] + \text{Mem}[C] \)
  - ISA equivalent
    - \( \text{Mem}[\text{EA}00] = \text{Mem}[\text{EA}08] + \text{Mem}[\text{EA}10] \)
- Low-level language equivalent
  - Hardware: ex. Machine (for a simple architecture)
    - Load R2, B
    - Load R3, C
    - R2 \( \leftarrow \) R2 + R3
    - Store A, R2
- Executed as one step in an overall algorithm to solve a problem on a general-purpose digital computer

Universal computing devices

- Turing’s Thesis: Computer scientists believe that ANYTHING that can be computed, can be computed by a computer (provided that it has enough time and enough memory).
- What does this imply?
  - All computers (from the least expensive to the most expensive) are capable of computing EXACTLY the same things if they are given enough time and enough memory.
  - Some computers can do things faster, but none can do more than any other computer.
- All computers can do exactly the same things!
- Thus, any given problem is either computable or it is not computable
  - Problems may be computable, but still not feasible (NPC)
- This course focuses on one computer (the LC-3), but the concepts apply to all computers!

Exponential growth

- \( 10^1 \)
- \( 10^2 \)
- \( 10^3 \): Number of students in the college of engineering
- \( 10^4 \): Number of students enrolled at Wright State University
- \( 10^5 \): Number of people in Dayton
- \( 10^6 \): Number of people in Ohio
- \( 10^7 \): Number of stars in the galaxy
- \( 10^8 \): Total number of particles in the universe
- \( 10^{100} \): Number of possible solutions to traveling salesman (100)
- Traveling salesman (100) is computable but it is NOT feasible.

CS Realities: Why study Computer Org?

- You’ve got to understand the limitations of binary encodings
  - Integers, floating point numbers (rounding, range, precision), strings of characters, instructions
- You’ve got to understand how a machine processes instructions
  - The stack
- You have to understand memory
  - Memory references (pointers)
- Computers do more than execute your program
  - I/O, interrupts, network behavior
- In short:
  - You need to understand (well) the limitations of the level of abstraction directly below your focus of effort.
  - "People who are more than casually interested in computers should have at least some idea of what the underlying hardware is like. Otherwise the programs that they write will be pretty weird."
    - - Donald Knuth, The art of programming