Computer science overview

- What is Computer Science?
  - “Computational thinking will be a fundamental skill used by everyone in the world by the middle of the 21st Century.” - Jeannette M Wing, Department of Computer Science, Carnegie Mellon
  - Approach to solving problems.
  - Designing Systems.
  - Understanding human behavior.

- What it is not.
  - Not the vocation based area’s of the computer field (MIS, IT, etc).
  - It isn’t Programming, web-browsing, and word-processing.
    - It is the theoretical foundation of information and computation.
    - Practical techniques for their implementation and application.
Low-level Computer Science: Deal with understanding of computer hardware, its internal operation and their peripherals.
- Processor design.
- Memory layout (local memory vs distributed memory for parallel systems).
- Network design.

High-level Computer Science: Deals with software mostly.
- Designing programming languages.
- How to program multi-cores systems.
- Distributed application development techniques.
- Efficient techniques for encoding sound or video (mp3, mpeg, divx, etc).

Common trend: Computer Science is a broad all encompassing field.
Basics: How it all fits together.
Describing the components - Hardware

- Central Processing Unit (CPU) - “Processor’.
  - Considered the “brain” of the computer.
  - What makes an AMD chip different from an Intel chip?
- Random Access Memory (RAM).
  - Thought of as a workspace: all storage here is temporary.
- Input/Output devices (I/O devices).
  - Allow human interaction with computer (not exclusively).
  - Monitors, Keyboard, Mouse.
  - Mass storage devices (CD-ROM, Blu-Ray Devices, Hard Drives, etc).
Describing the components - Software

- **Operating System.**
  - Provides an environment in which “processes” can run (execute their instructions): some allow multiple, concurrently.
  - Does resource allocation and management for running processes: interface between hardware and software.
  - Allows user interface.

- **Programs - Process**
  - Executes a single sequence of instructions in an *address space*.
  - A program counter (PC) in hardware keeps track of the instructions to execute.
  - With many processes active the OS provides some aspect of a *virtual machine*.
    - Run in a virtual address space: keep each process instructions separate.
    - Processes compete for resources such as processor, memory, and peripheral devices (managed by OS).
Machine Cycles

Central Processing Unit

- Control Unit
- ALU
- Registers
- Memory (RAM)

Cycle steps controlled by Clock

0000 Instruction
0016 Instruction
0032 Instruction
0048 Instruction
0064 Instruction
0080 Instruction
0128 Data
0144 Data

Machine Language
Understanding the machine

Components

- Memory (RAM) - where processes (programs) are loaded before execution: instructions and data.
  - CPU’s have L1 or L2 or L3 cache - a different type of RAM.
  - Sometimes on the chip sometimes off the chip.
- Registers - small working memory area on the CPU used to temporarily hold values.
- Arithmetic Logic Unit - Performs mathematical and logic operations.
- Control - Governs reading and writing of Registers and operations performed by ALU.

Machine Cycle

- Fetch - Load instruction from Main memory into Register.
- Decode - Determine the type of instruction and its operation.
- Execute - Perform operation from instruction.
- Store - Store any results to registers and ultimately Main Memory.
Basic Pipeline
What makes it tick?

Clock

- Controls the computer “steps” (cycles), which occur X times per second.
- Cycles are measured in Hertz (Hz): “cycles per second”.
- Each processor has a different clock cycle: AMD vs Intel, I7 vs Phenom II.

True or False

- A 3.0 GHz machine is always faster than a 2.5 GHz machine?
  * (Faster: completing a set of instructions the fastest) *
Understanding the machine language - Instructions

Machine Language

- What does a computer “instruction” look like?
- What language do we “talk” to tell a computer what to do? (Writing Programs)

Understanding Language

- Imagine creating your own code for the English Language.
- Assign a number to each word in the English Language (100,000+).
- How to make it simpler to remember for you and easier to translate?
  - Assign a number to each of the letters instead (26).
## Encoded Language

<table>
<thead>
<tr>
<th></th>
<th>THE</th>
<th>CAT</th>
<th>ON</th>
<th>THE</th>
<th>MOON</th>
<th>MAKE</th>
<th>Cakes</th>
</tr>
</thead>
<tbody>
<tr>
<td>THE</td>
<td>19</td>
<td>7</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>19</td>
<td>14</td>
</tr>
<tr>
<td>MOON</td>
<td>12</td>
<td>14</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>Cakes</td>
<td>2</td>
<td>0</td>
<td>10</td>
<td>4</td>
<td>18</td>
<td>17</td>
<td>8</td>
</tr>
</tbody>
</table>

* Note: all counting starts at 0 from now on.
Using the computer language

- Computer are made up of nothing more than switches.
- A switch has two states: On or Off.
- How many letters does our language have now?
  B  A  B  A  A  A  B  A  B  A
  1  0  1  0  0  0  1  0  1  0
- How useful is this? How do we get our 26 letters back?
Human Language vs Machine Language

Machine Language - Binary

- The numbering system using only zeros and ones is called **binary**.
- Refering to a single “letter” in this number system is called a **bit**.
- Patterns of bits can be used to represent our letters or any other pattern called **encoding**.
- Computer instructions are represented by their **instruction set**.
  - Collection of strings representing different instructions.

Human Language - High Level Languages

- To make communication easier humans use high level languages:
  ```c
  int main(int argc, char argv*[]){
      printf("Hello world!\n");
  }
  ```
- Use other programs to translate instruction into machine language called **compilers**.