Programming 1 - Honors

Lecture 1
COP 3014 Spring 2017

January 10, 2017

Main Components of a computer

- CPU - Central Processing Unit: The "brain" of the computer.
  - ISA - Instruction Set Architecture: the specific set of low-level instructions available to a CPU. Differs for various CPU types (Intel Pentium, Mac G4, etc).
- ALU - Arithmetic & Logic Unit responsible for performing arithmetic calculations, as well as logical operations (comparisons for equality, inequality, for instance).
- Main Memory (RAM - Random Access Memory).
  - storage close to CPU
  - Faster to access than hard disk
  - stores executing programs and data being currently worked on
- Secondary Memory
  - hard disk, floppy disk, CD, DVD, etc.

Main Components of a computer

- Input devices
  - mouse, keyboard, scanner, network card, etc.
- Output devices
  - screen/console, printer, network card, etc.
- Operating System
  - Examples: Mac OS, Windows XP, Linux
  - Controls computer operations
  - Manages allocation of resources for currently running applications

Memory Concepts

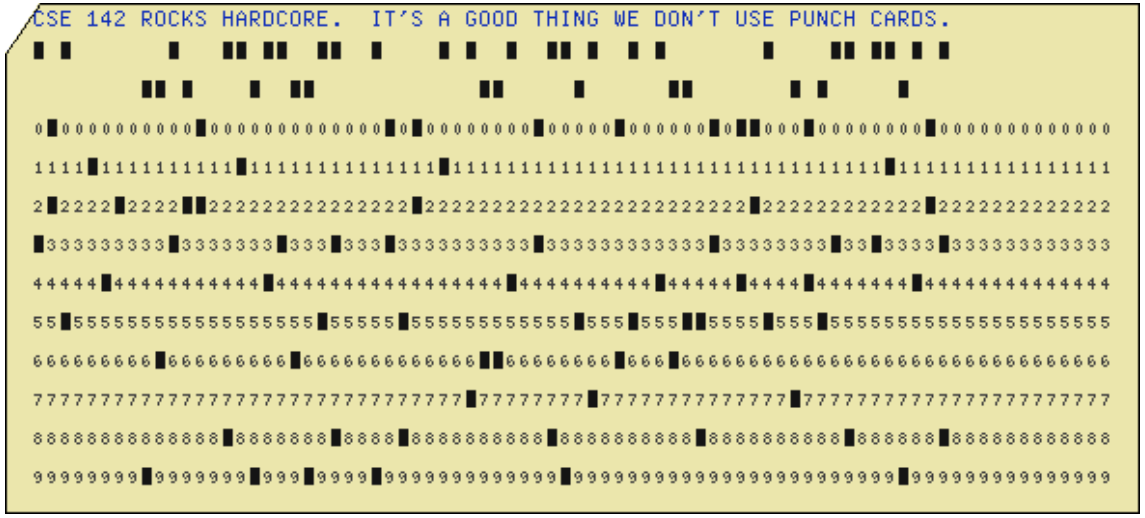
- bit: a binary digit
  - Stores the value 0 or 1
  - Smallest unit of storage in a computer
- byte: 8 bits
  - Smallest addressable unit of storage in a computer
  - Storage units (variables) in a program are 1 or more bytes
  - Each byte in memory has an address (a number that identifies the location)

Programming, and Programming Languages

Program - a set of instructions for a computer to execute

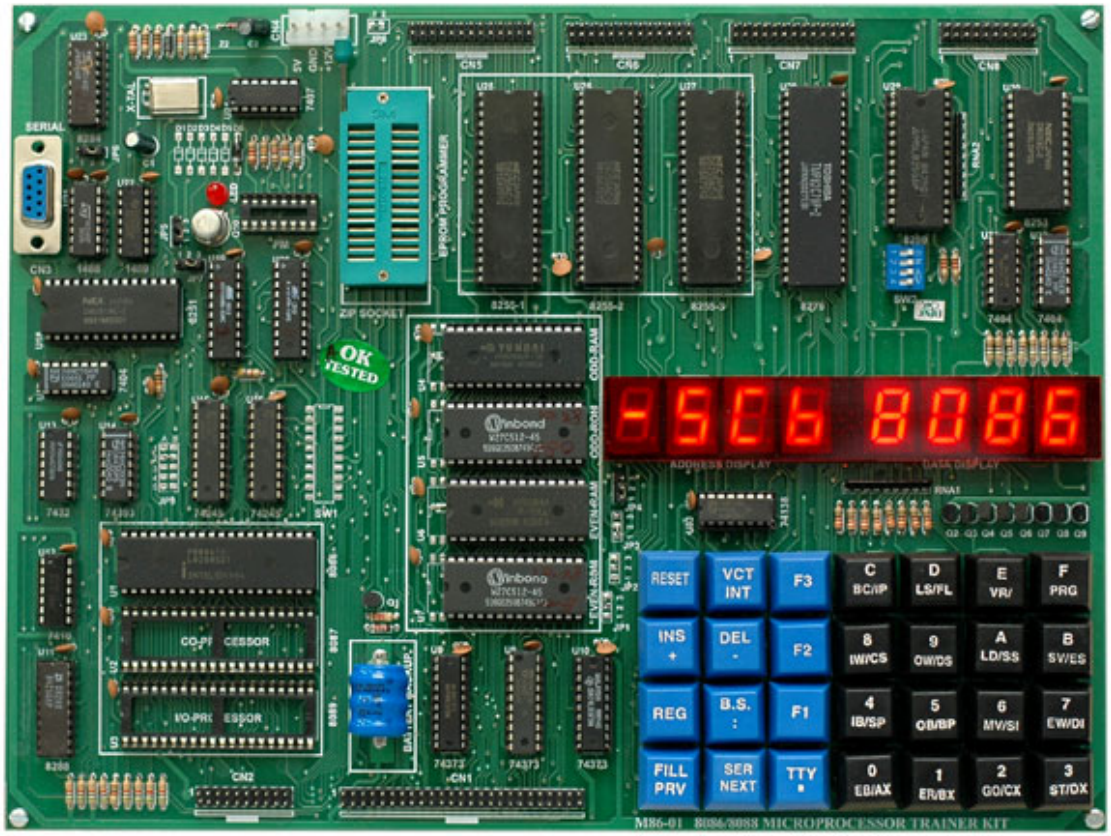
Evolution of Programming languages

- Machine Language
  - Based on machine's core instruction set
  - Needed by computer, hard for humans to read (1's and 0's)
  - Example: 11101101010101100011101010



Programming, and Programming Languages

- Assembly Language
  - translation of machine instructions to symbols, slightly easier for humans to read
  - Example: ADD $R1, $R2, $R3



Programming, and Programming Languages

- High-level procedural languages
  - Abstraction of concepts into more human-readable terms
  - Closer to "natural language" (i.e. what we speak)
  - Easy to write and design, but must be translated for computer
  - Examples include C, Pascal, Fortran
- Object-oriented languages
  - Abstraction taken farther than procedural languages
  - Objects model real-world objects, not only storing data (attributes), but having inherent behaviors (operations, functions)
  - Easier to design and write good, portable, maintainable code
  - Examples include Smalltalk, C++, Java

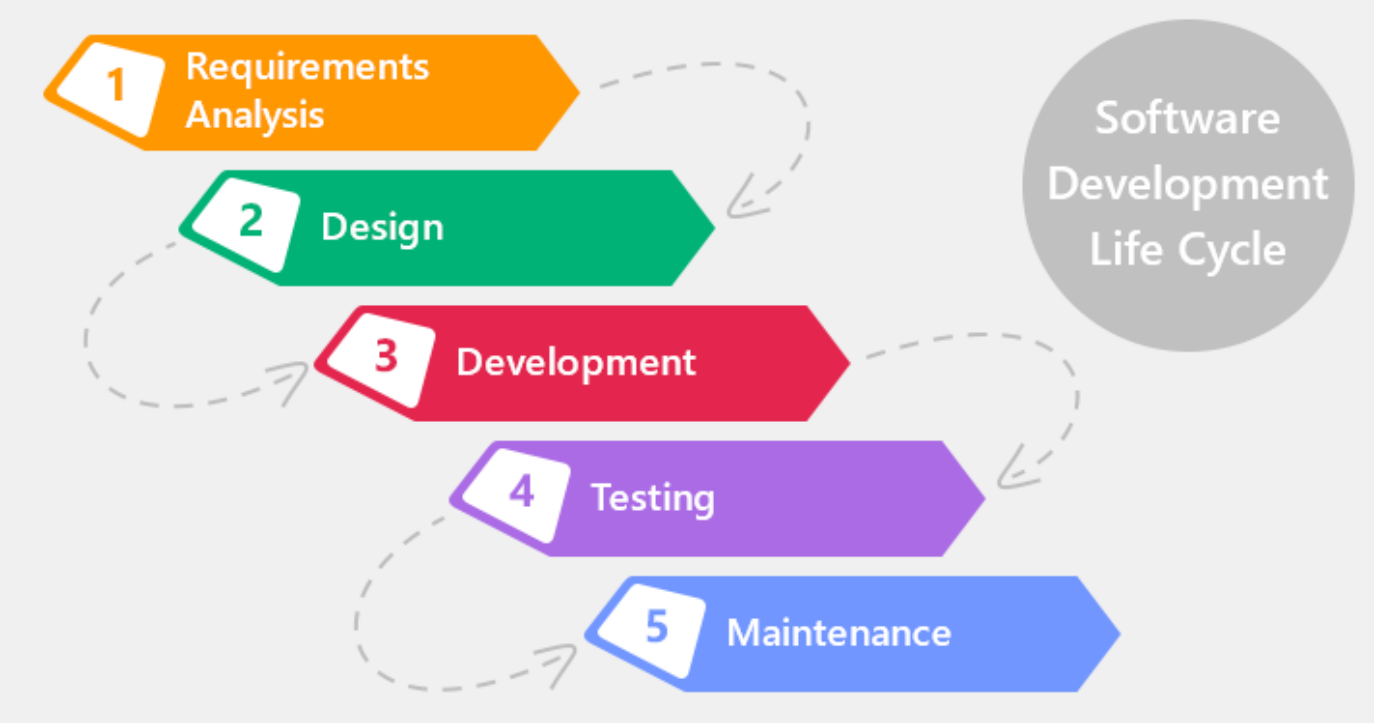
Code Translation

Bridging the gap between high-level code and machine code

- Interpreted languages – source code is directly run on an interpreter, a program that runs the code statements
- Compiled Languages
  - A compiler program translates source code (what the programmer writes) to machine language (object code)
  - A linker program puts various object code files together into an executable program (or other target type, like a DLL)
  - C and C++ are compiled languages

Software Development

Involves more than just writing code



Software Development

- Analysis and problem definition
- Design - includes design of program or system structure, algorithms, user-interfaces, and more
- Implementation (coding)
- Testing - can be done during design, during implementation, and after implementation
- Maintenance - usually the major cost of a software system. Not part of "development", but definitely part of the software life cycle

Programming is about Problem Solving

- Algorithm - a finite sequence of steps to perform a specific task
  - To solve a problem, you have to come up with the necessary step-by-step process before you can code it
  - This is often the trickiest part of programming
- Some useful tools and techniques for formulating an algorithm
  - Top-down Refinement: Decomposing a task into smaller and simpler steps, then breaking down each step into smaller steps, etc
  - Pseudocode: Writing algorithms informally in a mixture of natural language and general types of code statements
  - Flowcharting: If you can visualize it, it's often easier to follow and understand!

Programming is about Problem Solving

- Testing - algorithms must also be tested!
  - Does it do what is required?
  - Does it handle all possible situations?
- Syntax vs. Semantics
  - Syntax – the grammar of a language.
    A syntax error: "I is a programmer."
  - Semantics – the meaning of language constructs
    Correct syntax, but a semantic error: "The headphones ate the tree."

Basic Creation and Execution of a C++ program

- Create **source code** with a text editor, store to disk.
  - Source code is just a plain text file, usually given a filename extension to identify the programming language (like .c for C, or .cpp for C++)
- **Preprocessor** – Part of compiler process, performs any pre-processing tasks on source code.
- **Compilation** – syntax checking, creation of *object code.*
  - Object code is the machine code translation of the source code.
- **Linking** – Final stage of the creation of an executable program. Linking of object code files together with any necessary libraries (also already compiled).
- Execution of program
  - Program loaded into memory, usually RAM
  - CPU executes code instructions

Software Required for the Class

- A text editor compatible with C++. There are several free text editors.
  - We will be using **Notepad++** in class.
  - Sublime (available for Windows and Macs).
  - Atom.
  - Vim/Emacs for Linux.
- A File Transfer Client
  - Putty, BitVise or SSH Client for Windows.
  - Cyberduck or Filezilla for Macs.
- You can also use IDE's like Visual Studio or XCode. However, if you do so, you still have to test your code on `linprog` before turning it in.
- An account on the CS department programming servers.