Hardware: Central Processing Unit (CPU)

- The main electronic chip performing the computation
- The CPU usually performs very simple tasks
  - add/subtract/multiply two numbers at a time
  - compare two numbers to see which is larger
  - move/copy data from one spot to another
  - etc. . .
- The CPU *appears* powerful (intelligent?) *Why?*
  - Because it does all of these things *very quickly*

Software: Computer Programs

- Software is a collection of *instructions* and *data*
- A program tells the CPU what to do
- A program must therefore “speak the language” of the CPU, telling it what to do in terms of a combination of the simple operations it can actually perform
A programming language lets us:

1. Write down commands in a way that we can understand at a relatively sophisticated human level
2. Do so in a way that can be translated into “CPU-speak,” at the level of basic machine operations

A very great difference exists between those levels

- Many degrees of possible language complexity
- Each language looks different and works somewhat differently at the high level of the programmer
- However, each gets “translated” into the same simple language of the CPU in order to run the program

Compiling a Program

- Programmers write at the higher level of a programming language
- To make the CPU understand, it is translated to the lower, machine level, CPU Instructions
- The translator is itself a program, called a compiler
  - Since each system/CPU will have different ways of doing things, each has its own low-level “language”
  - The translation will only run on certain specific systems/CPUs
  - To get the same program to run on a different sort of machine, you have to compile (translate) it again

Levels of Languages

- High Level
  - Closer to Human Language
- Low Level
  - Close to CPU (add, subtract, etc.)

The Process in Java

- Things are slightly different here!
  - We have another program, the Java Virtual Machine (JVM)
  - The JVM translates Java bytecode into actual CPU instructions
  - In Java, the first compiler step actually translates our instructions into this bytecode (middle-level), which is a combination of human-readable and formal constructions
  - Advantage: the same bytecode can run on any OS for which a JVM exists (which does its own final translation). Platform independent.
  - Disadvantage: someone has to write the JVM for each OS (the same thing is true for compilers, however)
  - Another (partial/possible) advantage: the JVM also seeks to verify the bytecode to avoid dangerous bugs and security holes (due to how complex programs may be, this is only partially possible)
Compiling a Java Program

Remember: this isn’t really such a bad situation.
Before, we had to write a separate compiler for each platform, anyway.

Java Program

Compiler

Bytecode

JVM
- Intel Pentium (Windows)
- Intel Core i7 (Mac OS X)
- Atom D425 (Linux)

The Java Programming Model

Action | Tool | Result
--- | --- | ---
Programmer types in commands | Text Editor | Source code file (className.java)
Software is translated into computer’s “language” to be run | Java Compiler | Bytecode file (className.class)
The program executes. | Java Virtual Machine |

For This Week

- Meetings this week:
  - Monday, Tuesday, Friday: regular classroom
  - Wednesday in the CS Lab (16 Wing)
    - Important to be there to get oriented with the lab
    - You can bring your own computer if you like, but we will ensure everyone can log in to lab machines properly
- Please obtain the online text ASAP
- First reading assignment due: 12:00 PM, Thursday 06 February

- Office Hours: Wing 212
  - Monday/Wednesday/Friday: 11:00 AM–12:00 PM
  - Tuesday: None this week (normally 3:15 PM–4:15 PM)