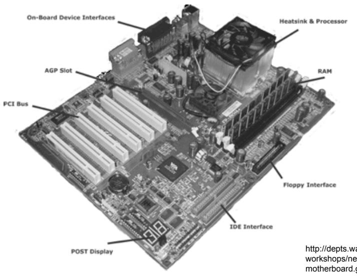
Introduction to Computer Programming

The process of programming

Parts of the computer



http://depts.washington.edu/sacg/services/workshops/networking/trouble_mac/img/motherboard.gif

Computer Instructions

- Computer programs are made up of individual instructions that are stored in RAM in binary format (machine code)
  - e.g. 01001010 00001111
    "add to" "register 15"
- Instructions may also require data to execute
  - e.g. Add **100** to register 15
    01001010 00001111
    00000000 01100100 (100_{10} in binary)

Computer Instructions

- Each byte has an associated memory address.

Address	Contents	
200	01001010	one computer instruction
201	00001111	
202	00000000	
203	01100100	
204	01011011	another instruction
205	01100101	

Executing a Program

- To execute a program, we just have to "tell" the CPU which address has the first instruction, and it goes from there.
  - The operating system (Windows, MacOS) has an application called a loader that loads a program into memory so that the CPU can execute it.
- How do we know what binary values make up a program we want the computer to execute?
  - This is where programming languages like Java come in!

Programming Languages

- High-level programming languages allow us to write programs so that they are readable to us.
- A compiler is a program that translates our high-level language program to machine language for execution on a computer.
- Some high-level languages:
  C++ C Fortran Java Ada

Typical Compilation

`z = x + y;` ← Assignment statement in a high-level language

Compiler translates this instruction to the machine code:

10010101 01100001	(load x into reg 1)
10100110 11010001	(add y to reg 1)
10111100 01010001	(store reg 1 into z)

Problem: These machine instructions only work for the specific CPU for which the compiler is designed.

Java Compilation

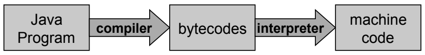
`z = x + y;`

Compiler translates this instruction to a "virtual" machine code called "byte codes":

0001 1010	(iload_0)
1010 1011	(iload_1)
0110 0000	(iadd)
0011 1101	(istore_2)

Java Execution

The byte codes then can be executed by using a Java interpreter (JVM - Java virtual machine) that translates each byte code to the machine code for the specific CPU on which the program is to be executed.



Now the Java bytecodes can be run on ANY computer as long as the computer has an interpreter.

Goals of the Course

- Learn the <u>syntax</u> of the Java language
  - Syntax refers to the rules of the language (punctuation, naming variables, valid structure of statements, etc.)
- Learn how to put sequences of instructions together to form programs that solve useful problems.
  - Also, learn how to "debug" your programs when they don't do what you thought they would do!

Types of Programming Errors

- Syntax errors
  - Writing a statement that doesn't follow the rules of the programming language.
  - Example: **`x + y z =;`**
  - A compiler cannot translate statements with syntax errors into machine code (or bytecodes).
    - Therefore, you cannot run a program with syntax errors in it.

Types of Programming Errors

- Logical Errors
  - Writing a statement that follows the rules of the language but doesn't do what you intend.
  - Example: **`z = x * 2;`**
    (You meant to write **`z = x + 2;`**)
  - Compiler can translate this instruction into binary, so program can be executed.
    - But z will get the wrong answer during execution.
      ...except when?

Types of Programming Errors

- Runtime Errors
  - Writing a statement that follows the rules of the language but causes a program failure during execution.
  - Example: `z = x / y;`
  - Compiler can translate this instruction into binary, so program can be executed.
    - But what happens if y stores the value 0 as this instruction is executed?

Your mission, if you choose to accept it...

- Write computer programs that contain none of these errors!
- Knowing this skill will benefit you, no matter what discipline you're in!