Introduction

CSE 142, Summer 2002

Computer Programming 1

http://www.cs.washington.edu/education/courses/142/02su/

Why Are We Here?

- Computers are everywhere!
  - Big ones serving databases and forecasting the weather
  - Medium sized computers on your desk top, for playing games, writing papers, surfing the internet
  - Tiny ones everywhere: cars, microwaves, toys, phones
- They're part of our world
  - What can they do? How do they do it?
- What can *you* do using a computer program that you've written yourself?

Two Interesting Facts

1. Computers are multi-purpose
   - Unlike cars, toasters, dishwashers
   - The same physical computer can play games, solve equations, plan trips, send e-mail, etc. How is this possible??
   - Answer: the computer operates under direction of a "program": a set of precise instructions
2. The largest and the smallest computers have much in common
   - We can usefully think of about computers in general without worrying about hardware details
   - This is our first example of "abstraction", a key notion in computer science

Computers & You & CSE142

- You'll learn to write programs
  - We use a particular language called Java
  - The principles apply to many other languages
- You'll use particular computers
  - Windows, Mac, Linux, Unix, whatever
  - Principles apply to many computers and operating systems
- We'll talk about the process of software development
- Useful class if you want to understand how computer systems are developed and how they operate
  - programmer, technical user, business user, manager, purchaser, …

What To Expect

- Course is for beginners
- Programming is quite different from using applications
  - Logic/problem solving skills
  - Can be challenging, but also very rewarding
- Important to keep up
  - Ask for help when you need it; don't fall behind
- If you have the background to skip this course, you can go directly to CSE 143
  - Automatic credit for CSE 142 after completing CSE 143 successfully

Readings and References

- Reading
  - Chapter 1, *An Introduction to Programming and Object Oriented Design using Java*, by Niño and Hosch
- Other References
  - All course information is available from the web site:

    http://www.cs.washington.edu/education/courses/142/02su/

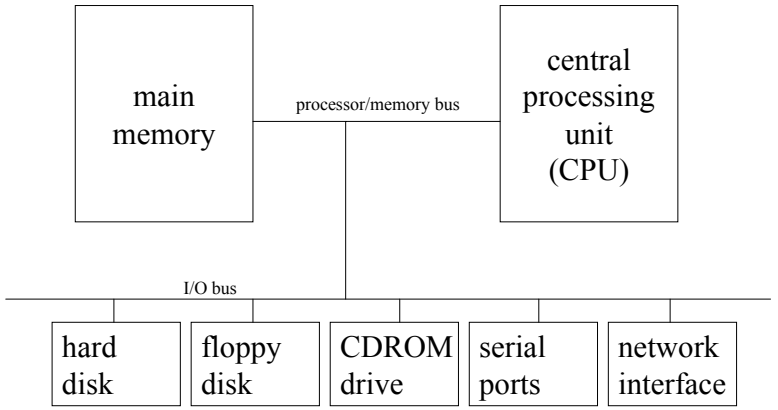
Hardware: the physical machine

- Central Processing Unit (CPU)
  - Pentium, PowerPC, SPARC, ARM, ...
- Memory / Random Access Memory (RAM)
  - main memory
- Hard disk, floppy disk, CDROM
  - disk storage
- Monitor, speakers, keyboard, mouse
  - input / output (I/O)
- Network connection
  - modem or Local Area Network

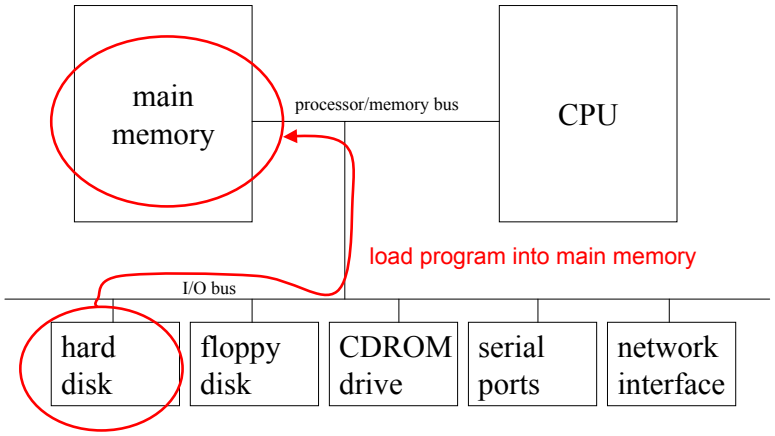
Software: the personality

- Software
  - the plans that instruct the hardware what to do
  - software defines the personality of the system
- Hardware doesn't do much on its own
  - But one set of hardware can do many different things if given the right instructions
  - It is often easier to update software than hardware

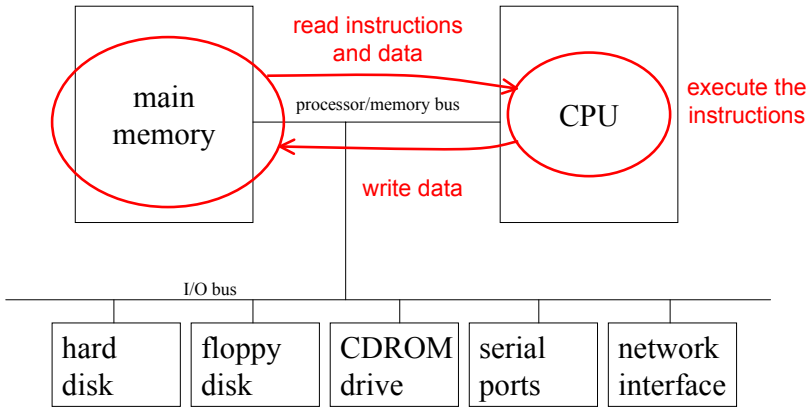
A typical computer organization



Running a program - program load



Running a program - program execution



Learning Programming

- Programming is both easier and harder than most people make it out to be.
  - Easier: Many of the things good programmers do well are actually things all of us already do all the time, we just don't know it.
  - Harder: Programming is in large part a *skill*, even an *art*
- Programming is like any craft: it requires practice.
  - Learning by doing vs. learning by reading about it
  - Not sure how something works? Try it and see!
  - Build things and throw them away. Experiment!
  - Don't be afraid that you will break the computer.

    get a cheap used one ($20) and tear it apart … it's fun!

Programming as Communication

- When we write a program, we are communicating with
  - the computer
  - other people
- The computer reads our program as the set of instructions that it should perform
  - It just needs to know how, not why
- Other people read our programs to understand how and why
  - Programs that don't work (bugs)
  - Program evolution - new features
  - Performance improvement
  - Project completion (you never did finish all those features last year …)

Communicating with Computers and People

- Computers need precision and logical thinking on your part
  - Being precise, complete, and logical is one thing that makes programming hard
  - Computers offer speedy, by-the-book results
  - What can go wrong with by-the-book results? No "common sense"!
- People can fill in missing steps, but can get swamped by lots of unorganized details and clutter
  - Need to write programs so that can be understood by people, e.g., your coworkers, your clients, yourself 3 months from now
  - Invent *abstractions*: new vocabulary, short-hands
  - Be *organized*, use good *style*

Example: Giving Directions

- Imagine giving campus directions:
  - To another student
  - To a tourist
  - To a robot
- The student operates at a higher level of *abstraction* with a richer *vocabulary* of *short-hands*
- An *algorithm* is a plan for how to accomplish a task
  - A *program* is a software implementation of an algorithm
- Good algorithms (at any level of abstraction) require precision

Metaphor: Programs as Directions

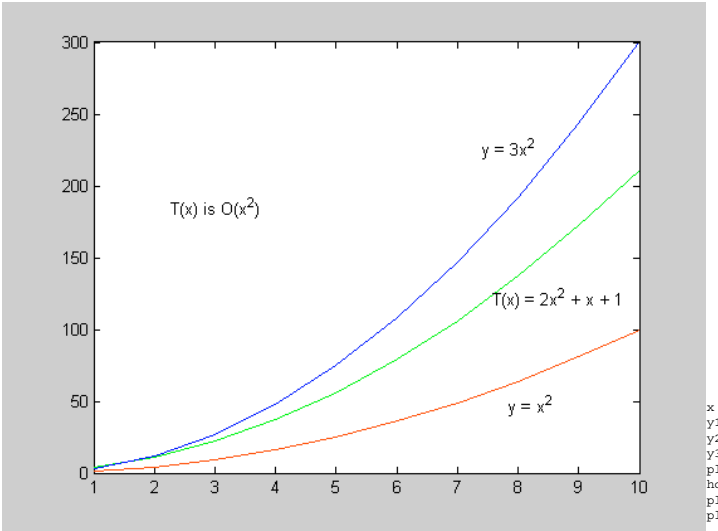
- One way to think about programming:
  - a program is a sequence of commands that brings about some action
- telling a robot how to navigate around campus
- telling a human visitor how to get from here to there
- telling a student (a higher form of human) how to get from here to there

Metaphor: Programs as Math

- We also can think of programs as *executable math*: a program calculates some result for us.
- Consider:

$$Area = \pi \cdot Radius^2$$

- We can employ such expressions in programs.
- Most of our intuitions and knowledge about mathematics apply to computers.




```
x = 1:10;
y1 = x.*x;
y2 = 2*(x.*x)+x+1;
y3 = 3*(x.*x);
plot(x,y1,'r')
hold on
plot(x,y2,'g')
plot(x,y3,'b')
```

Using the program Matlab to calculate and plot function values

Metaphor: Programs as Simulations

- We also can think of programming as creating or simulating both real *and virtual* worlds.
- We can define things in our programs that model the things in our world. We call these things *objects*.
- Programs are *plastic*: they are easy to mold to our wishes
  - Can be free of the constraints of real life!
- The limit of plasticity: big programs become as hard to work with as real-world entities

```
bird.exe
```