Anjuli Kannan

Reading Binary Numbers (2:00-7:30)

- Bring 8 volunteers down and have them represent the 128s, 64s, 32s, 16s, 8s, 4s, 2s, 1s places
- Volunteers raise hands if they represent a 1
- Students must answer: What decimal number do they represent and what ASCII letter is it?
- Round 1: 01000010 → 66 → B
- Round 2: 01001111 →79 → O
- Round 3: 01010110 →87 → W
- The number of bits we are using determines how many different numbers we can represent
  - 8-bit value can represent 256 (2^8) numbers
  - 16-bit value can represent 65,536 (2^16) numbers
  - 24-bit value can represent 16.7 million numbers
  - 32-bit value can represent 4 billion numbers
- The same basic building blocks that we use to represent numbers from 0-255 can be used to represent numbers up to 4 billions

Using Binary to Represent Negative Numbers (7:30 – 21:00)

- But how can we represent negative numbers?
- First way—use the leftmost bit to indicate negative
  - 1011 means -3 and 0011 means +3
  - Comes at cost of not being able to represent as many positive numbers
  - This is called the signed magnitude approach
  - We will see in C that we can indicate whether we are using signed or unsigned integers, and the computer will know how to interpret the bits accordingly
- Second way—flip the bits (0s becomes 1s and 1s becomes 1s)
  - 0011 means +3 and 1100 means -3
  - This is called ones' complement
  - Math is a little different, but still works out
- Third way—flip the bits and add one
  - 0011 means +3 and 1101 means -3
  - This is called twos' complement
  - Now we do addition like normal:

```
   4     0100
+ -3   + 1101
------  -----------
   1     0001   (we throw certain carries away)
```

- Using twos' complement means that addition and subtraction are performed the same way, which makes life easy for the hardware (and the people who make the hardware)

Classification of Programming Languages (21:00-32:00)

- When you run a Scratch program, your program is not being fed directly to the CPU
- Rather, it is fed into a virtual machine (VM), a software that can interpret Scratch
- CPU is underneath the VM
- Underneath the hood, Scratch is implemented in another language
- Java, too, is run by a virtual machine that you must download if you want to run Java programs
- That way, you can run it on any type of computer and it will run the same way—it's platform-independent
- C and C++, by contrast, are compiled languages
- That means that we write C source code, feed it to a compiler, get back an executable piece of object code, and run the object code
- The object code depends on the CPU and OS, so object code that we make in a Unix environment will not run in Windows
- If we want to execute our program in Windows, we would have to recompile it with a compiler that makes Windows compatible object code
- This is why you can't run your PC software and run it on your Mac
- In addition to compiled languages (C and C++) and languages that run on a virtual machine (Java) there are also interpretive languages like Perl, PHP, Python

Working in Linux (32:00-45:00)

- We will do all of our programming in a Linux (Unix) environment
- The New Instructional Computing Environment (NICE) is a bunch of computers locked up in the basement of the science center on which we will work
- You will use them remotely using software such as Secure CRT (Windows) or Terminal (Mac)
- After downloading this software, you can connect to nice.fas.harvard.edu
- What you will see is a command line interface – you can't do anything with your mouse
- First, configure your account for this course as explained in ps0
- You will initially be in your home directory (your personal space for coursework, email, etc.)
- You can use these commands
    - ls – list contents of current directory
    - pwd – print working directory
    - mkdir *dir* – make directory called *dir*
    - cd *dir* – open *dir*
    - nano *filename* – open *filename* in nano
    - cp – copy
    - mv – move
    - rm *filename* – remove *filename*
    - man *keyword* – look up *keyword* in manual

How to Write a Program in C (45:00-48:00)

- Write and save the following program as hello1.c

```
#include <stdio.h>

int
main(int argc, char * argv[])
{
      printf("hello, world!\n");
}
```

- At command line, type `gcc hello1.c` to make an executable
- Executable gets put in a.out
- Type `a.out` to run and we get the following result:

```
hello, world!
```

- But what if we want to give it a better name?
- Compile with this command: `gcc –o hello1 hello1.c`
- Now we can run by simply typing `hello1`