CMSC 216
Introduction to Computer Systems
Lecture 1
Introduction

Jan Plane & Alan Sussman
{jplane, als}@cs.umd.edu

Administrivia
• Course home page is at http://www.cs.umd.edu/class/fall2010/cmsc216
• If you don’t already have a GLUE account, request one at http://www.oit.umd.edu/new/
• Bring your laptop to discussion section on the next two Wednesdays and - after that – Every Monday!
• Read Chapter 1 of Bryant and O’Hallaron, and Chapter 1 of Reek

Introduction to Computer Systems
• Course objectives
• Expectations
• Course policies
• Discussion sections
• Course projects
• Submit server
• Grades server

Chapter 1, Bryant and O’Hallaron
A TOUR OF COMPUTER SYSTEMS
Storage of Information

- Computers store all data as binary digits, or bits; groups of 8 bits are often called bytes.
- How these bits are treated depends on their context:
  - the same sequence of bits can be used to represent a character, or an integer, or a floating-point number, or an instruction, or...
  - it's all a matter of interpretation.

Instruction-based execution

- Each program on a computer is a sequence of instructions written in machine language.
- Processor executes one instruction at a time in a program, then executes the next one in turn.
- To study code in this form, it's helpful to use assembly language rather than machine language code.

Example assembly program

```
main:    mov #0,sum    ; set sum to 0
         mov #1,num    ; set num to 1
loop:    add num,sum  ; add num to sum
         add #1,num   ; add 1 to num
         ble num,#1000,loop ; if num <= 1000, go back to 'loop'
halt
```

This is a slightly modified version of the example in Wikipedia's Computer article.

- What does this program do?
- Sequence of operations doesn't always go to the next instruction in memory.

Computer layout

- Lots of places to store information:
  - CPU registers
  - CPU caches
  - Main memory
  - Hard drives
  - Remote storage
- The farther away from the CPU you go, the longer it takes to access data.
- Typical programs have to access data stored on a hard drive, which is quite slow compared to other storage mediums.
Caching is important

• Executing a program can mean reading instructions from disk into memory, then moving around data from memory to registers or memory to disk
• Because some devices are much slower (maybe because they're bigger), we can utilize caches to speed up execution time by accessing copies of data
• This can be a major performance gain - properly utilizing caches can increase performance by orders of magnitude

The role of the operating system

• Protect the computer from misuse
• Provide an abstraction for using the hardware so that programs can be written for a variety of different hardware
• Manage the resources to allow for reasonable use by all users and programs on a computer

The UNIX Operating System

• Developed in 1970s at Bell Labs
• Kernel written in C, also developed at the same time
  – C was developed for the purpose of writing UNIX and systems programming
• We will use a variant of UNIX named Linux
  – Do not try working in a Windows environment just because you're more comfortable with it!
  – Other UNIX variants exist, such as Solaris, and the various BSDs (OpenBSD, NetBSD, FreeBSD, OSX)

Processes

• Programs are often written as if they are the only things running on a system
• The OS allows them to work this way by providing an abstraction known as a process
• Process is a running program (one or more threads of control), along with all the data associated with it (an address space)
• OS uses context switching to give the appearance of multiple processes executing at once on a single processor
Virtual memory

- Each process is presented with the appearance of having 4 GB of available memory (on a 32-bit system) - this is virtual memory
- Physical memory ≠ virtual memory
  – Computer may not even have 4 GB of memory!
- Memory is organized in a particular manner; from bottom to top (in terms of addresses):
  – program code and data
  – heap
  – stack

Files in UNIX

- A file is a sequence of bytes - not a magical container holding the bytes, but the bytes themselves
- In UNIX, all I/O devices are modeled as a file
  – input from keyboard
  – output to screen
  – input/output from/to disk
  – input/output from/to network port
- Specific details of file organization can vary from OS to OS, and even filesystem to filesystem

Why learn about computer systems?

- Getting your programs to work correctly requires an understanding of how the computer does its work
- Making the computer do what you want can require in-depth knowledge of the OS
- For example, this Java method runs incredibly slowly, and it's entirely the programmer's fault:
  ```java
  public static int sumByColumns(int[][] array) {
      int sum = 0;
      for (int j = 0; j < COLS; j++)
          for (int i = 0; i < ROWS; i++)
              sum += array[i][j];
      return sum;
  }
  ```