Lecture 1

Overview

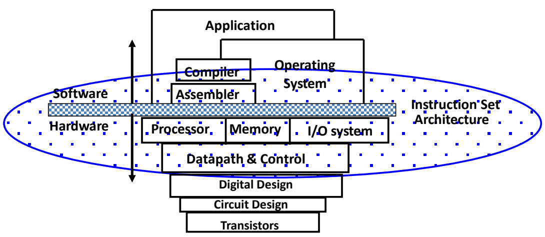
CPSC 275
Introduction to Computer Systems

What Really Happens When Running a Program?

```
public class Hello {
   public static void main(String[] args) {
      System.out.println("Hello, CPSC275!");
   }
}
```

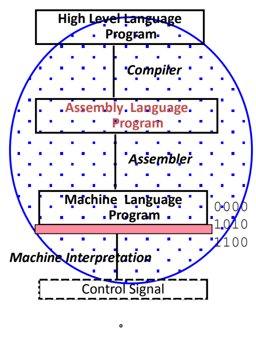
```
void main(int argc, char *argv[])
{
  printf("Hello, CPSC275!");
}
```

A Computer System



- Coordination of many *levels of abstraction*

Levels of Representation

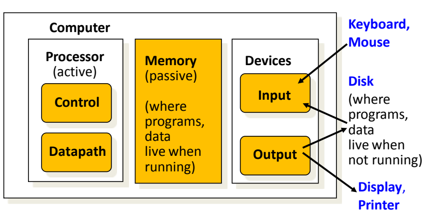


temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;

pushl %ebp
movl %esp, %ebp
popl %ebp

0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001

Five Major Components

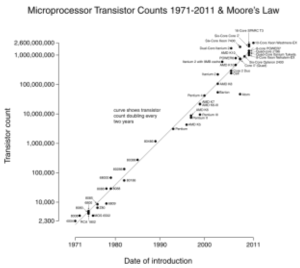


Dramatic Change in Computer Technology

- Processor
  - 2x in speed every 1.5 years
  - 100x performance in last decade
- Memory
  - DRAM capacity: 2x / 2 years; 64x size in last decade
  - Cost per bit: improves about 25% per year
- Disk
  - capacity: > 2x in size every year
  - Cost per bit: improves about 100% per year
  - 120x size in last decade

Moore's Law

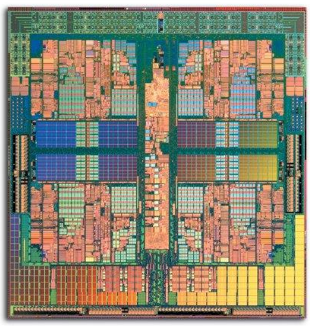
"The number of transistors on a processor chip will roughly double in every two years."



Hitting the Wall

- Limitations in device physics
- Key Issues
  - Higher clock speed (frequency)
  - Power consumption
  - Heat dissipation
  - Memory gap
- Solution: Multicore processors

A Typical Quadcore Processor



CPSC 275: So What's In It For Me?

- Understanding computer systems from a programmer's view
  - What the programmer writes
  - How it is converted to something the computer understands
  - How the computer interprets the program
  - What makes programs go slow

Big Ideas in CPSC 275

- 5 major components of a computer system
- Data can be anything (integers, floating point, characters): a program determines what it is
- Stored program concept: instructions just data
- Principle of abstraction, used to build systems as layers
- Principle of locality, exploited via a memory hierarchy (cache)
- Greater performance by exploiting parallelism
- Principles/pitfalls of performance measurement

Teaching Staff

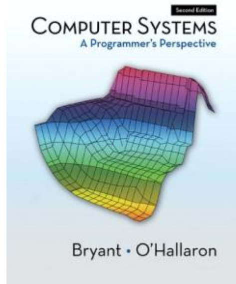
- Prof. Peter Yoon
  Office: MECC 127
  Phone: 297-2461
  Email: peter.yoon@trincoll.edu
  Office Hours: MWF 10:00-11:30 a.m.
- Teaching Assistants:
  - Philip Cho, '15
    Email: hyunsu.cho@trincoll.edu
    TA Sessions: TR 8-10 p.m.
    
  - Reid Delaney, '16
    Email: reid.delaney@trincoll.edu
    TA Sessions: MW 7-9 p.m.
    

Course Website

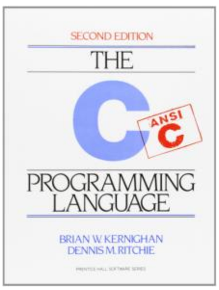
turing.cs.trincoll.edu/~pyoon/cpsc275

Make sure to register today!

Required Text



Strongly Recommended



Recommended

Lab

- M 1:15 – 3:55 p.m. in MECC 136
- Linux OS
- Shell programming
- Systems programming in C
- Debugging strategies
- Code optimization
- Code review
- And much more

Quizzes

- Weekly
- No make-up
- Will drop the lowest score

Exams

- Exam I at 1:15 p.m. on Sep 17
- Exam II at 1:15 p.m. on Nov 21
- Final Exam at 3:00 p.m. on Dec 11

Grading

Quiz	10%
Lab	10%
Exam I	15%
Exam II	15%
Assignments	30%
Final Exam	20%

Course Policy

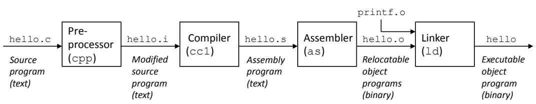
- Attendance
- Reading ***before*** coming to class
- Late work will ***not*** be accepted
- Learning disability
- Academic honesty

A Grand Tour of a Computer System

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

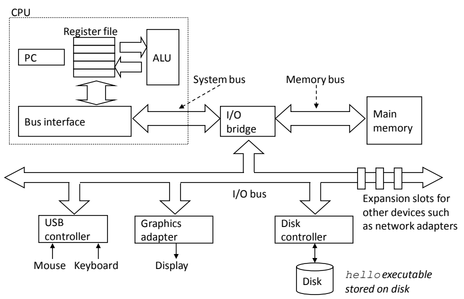
Program Translations



Computer Organization

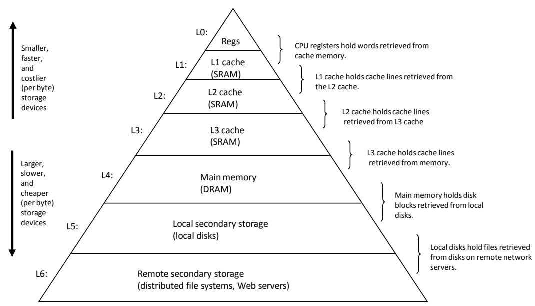
- Lots of places to store information:
  - CPU registers
  - CPU caches
  - Main memory
  - Hard drives
  - Remote storage
- The farther away from the CPU you go, the longer it takes to access data.
- Typical programs have to access data stored on a hard drive, which is quite slow compared to other storage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

Computer Organization, cont'd



What happens when running a program?

Memory Hierarchy



Cache Matters

- 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, 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.

An Example

Assuming the cache size = 1024*4 bytes, which of the following runs faster?

```
int a[1024][1024];
int row, col;

for (row = 0; row < 1024; row++)
   for (col = 0; col < 1024; col++)
     a[row][col] = 10;
```

```
int a[1024][1024];
int row, col;

for (col = 0; col < 1024; col++)
   for (row = 0; row < 1024; row++)
     a[row][col] = 10;
```

The Role of OS

- *Protect* the computer from misuse.
- *Abstract* 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.

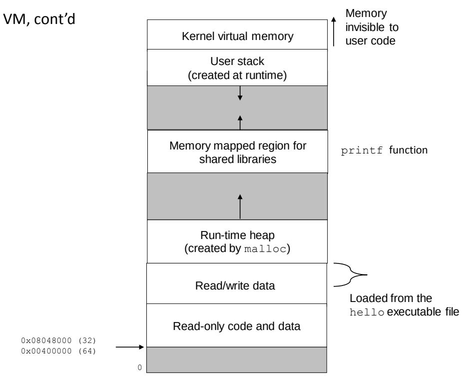
Processes

- Process is a running program, along with all the data associated with it (an *address space*).
- The OS allows them to work this way by providing an abstraction known as a process.
- Programs are often written as if they are the only things running on a system.
- OS uses *context switching* to give the appearance of multiple processes executing at once on a single processor.
- Can consist of multiple execution units called *threads*.

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
- Memory is organized in a particular manner; from bottom to top (in terms of addresses):
  - program code and data
  - heap
  - stack

VM, cont'd



Files

- A file is a sequence of bytes - not a magical container holding the bytes, but the bytes themselves
- In Linux, 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 Use C?

- Intermediate-level language:
  - Low-level features like bit operations
  - High-level features like complex data-structures
- Access to all the details of the implementation
  - Explicit memory management
  - Explicit error detection
- Better performance than Java/Python
- All this make C a far better choice for systems programming.

Come to C Workshop!

- Tonight 7-9 p.m. in MECC 136 or
- Tomorrow 8-10 p.m. in MECC 136
- Bring your laptop if you'd like install Linux OS on it.