# Practical Programming Methodology
(CMPUT-201)

Michael Buro

Lecture 1

- Introduction to the course
- Computer architecture



## General Course Information

- Section home page:
  - `www.cs.ualberta.ca/~mburo/courses/201`
  - news, schedule, lecture notes, and additional material
- Course news group:
  - ualberta.courses.cmput.201
  - You can ask general 201 questions here
- My E-mail: mburo@cs.ualberta.ca
- Office: Athabasca Hall 3-37
- Office Hours: Wednesdays 2-2:30pm or by appointment



## Course Work

- 11 Assignments (weekly) 33%
- Midterm exam (Feb. 16) 21%
- 12 Labs (Lab exam in week 11) 10%
- Final exam (TBA) 36%

Average weekly workload you can expect:

≈5h Lectures + ≈3h Lab + ≈1-3h Assignment = ≈9–11h

Final grades: 4-point scale, distribution method



## Lectures

- Attending is essential!
- Notes will be posted on course webpage after lectures
- Lecture notes of the W2005 course are available
- Differences: more emphasis on UNIX, tools, and C — less on advanced C++ features

## Assignments

- Crucial: Deepen the understanding of lecture material!
- 5–7 small-medium problems/programming exercises each
- Released weekly on Tuesdays
- Solutions have to be handed in by the following Tuesday 12:30p in class and/or electronically using the "astep" system
- Will be marked by TAs and discussed in the labs a week later

Solving assignments problems individually is the best way to learn!

Exam questions will be similar



## Labs

- Important!
- TAs present assignment solutions
- Mini-tutorials given by TAs
- Unmarked hands-on exercises, TAs help
- Lab exam in week 11 will test your ability to write and debug C/C++ code

Apply for UNIX account in CSC-143 this week

Labs start next week

Exams and assignments will also cover lab material



## Collaboration and Cheating Policy

- Discussing assignments among students is allowed! Programming is a team endeavour after all!
- Students must submit individual solutions and be able to explain their solutions.
- All sources — including books, webpages, and names of fellow students who took part in assignment discussions — need to be stated. Failure to do so constitutes plagiarism
- We use various plagiarism detection tools to compare submitted assignment solutions



## Software Engineering Courses

- 201: Small-scale programming
  - learn about UNIX/C/C++ and software libraries
  - get familiar with software development tools
  - know what goes on "under the hood"
  - design and implement interfaces and small programs
  - learn to appreciate software testing and defensive programming
- 301: Team work, object-oriented design
- 401: Large-scale programming

## CMPUT-201 Topics

1. The UNIX operating system, tools [1.5 weeks]
   - computer architecture, file system, commands
   - shell, text editor, customizations
2. Procedural Programming (C/C++) [5.5 weeks]
   - simple types, flow control, functions
   - arrays, pointers, structs, memory management
   - compiler, makefiles, debugger, profiler
3. Object Oriented Programming (C++) [3 weeks]
   - classes, operator overloading
   - inheritance
4. Generic Programming (C++) [2.5 weeks]
   - templates
   - Standard Template Library (STL)



## How to succeed in CMPUT-201?

- "Learning by doing"
- Don't hesitate to play around – it's hard to do any permanent damage if you create backups or use a version control system
- Write small programs to test new concepts
- Learn to find answers for yourself
  - Textbooks
  - man and web pages
  - Google compiler error messages
- Learn to use a debugger

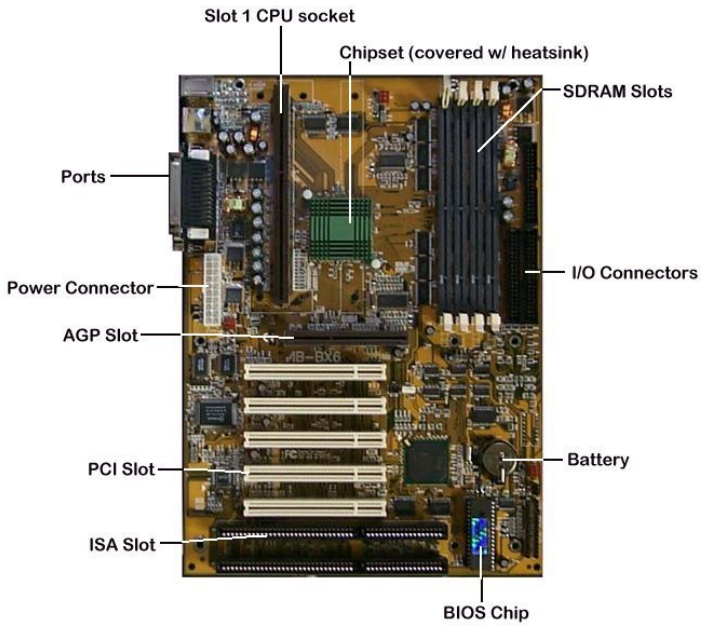


## How to fail/drown in CMPUT-201?

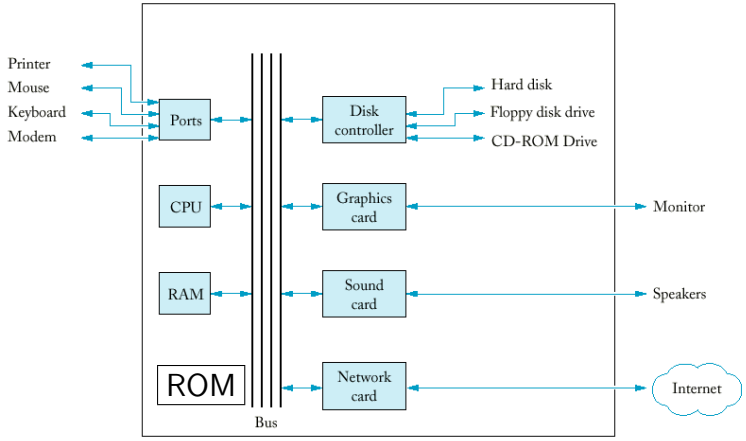
- Skipping lectures or labs
- Ignoring assignments or copying fellow students' solutions
- Not taking advantage of asking questions in labs
- Starting with programming prior to thinking about the problem and trying to make programs work by applying random changes
- Wasting considerable time by not learning how to use a debugger



## Typical PC Mainboard



## Schematic Design of a Personal Computer

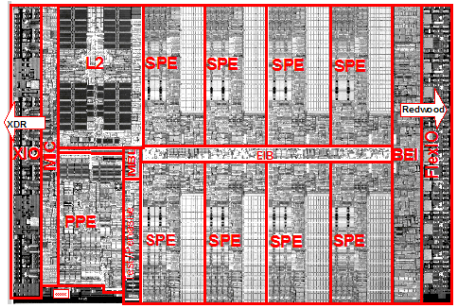



CPU = Central Processing Unit
ROM = Read-Only Memory

RAM = Random Access Memory

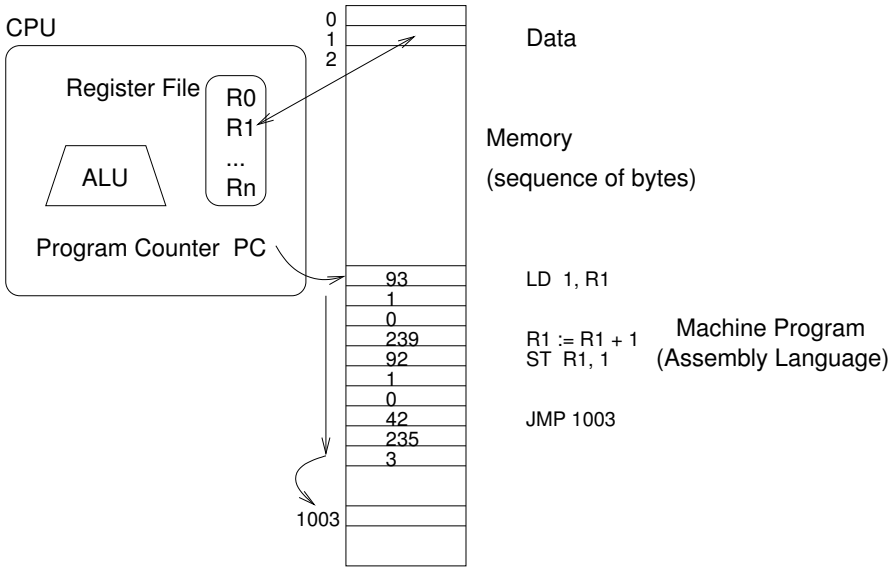


## Central Processing Unit (CPU)

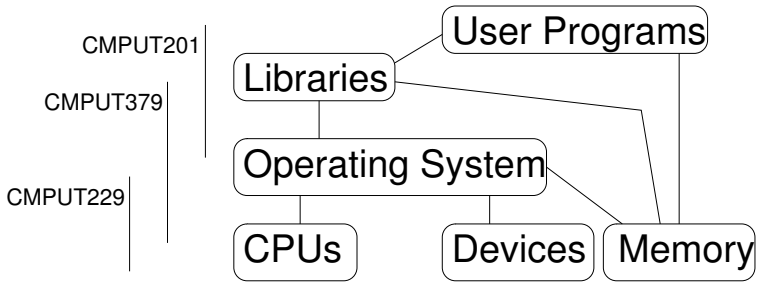



## von Neumann Architecture






## Software/Hardware Layers



## Part 1: The UNIX Operating System, Shell, Editor

Why UNIX?

- Open standards (e.g. POSIX threads)
- Dominant server operating system
- Free versions available (FreeBSD, OpenBSD, Linux)
- Many free software development tools:
  gcc, emacs, gmake, gprof, gdb, kdevelop, etc.
- Multi-tasking (multiple programs can run at the same time) / multi-user (multiple users can work on one machine) operating system
- We will be using GNU/Linux in the labs



## GNU/Linux

I highly recommend to administer your own Linux system at home. There are many freely downloadable distributions. E.g.

- Redhat Fedora Core
  `www.redhat.com/fedora`
  Frequent updates. Stable.
  Requires one or two partitions on your harddisk.
- Knoppix live CD/DVD
  `www.knopper.net/knoppix/index-en.html`
  Does not require any changes in your setup!
  Great for checking Linux out.