CSC 211: Object Oriented Programming

Introduction

Marco Alvarez

Department of Computer Science and Statistics
University of Rhode Island

Fall 2019

Team

- Instructors
  - Marco Alvarez
  - Michael Conti
- Graduate TAs
  - Christian Esteves
  - Yana Hrytsenko
- Undergraduate TAs
  - John Bertsch
  - Eben Aceto

Welcome !

- Lectures
  - TR 12:30 - 1:45p @ Beaupre 100
- Labs
  - F 12 - 1:45p @ Library 166
  - F 2 - 3:45p @ Library 166
  - M 2 - 3:45p @ Tyler 55
- Discussion Sections
  - M 5p - 6p @ TBA
  - T 5p - 6p @ TBA
- Office Hours
  - TBA

Typical Schedule

TIME	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
10:00 AM					
11:00 AM				Office Hours Mike 11AM - 12PM	Office Hours Marco 11AM - 12PM
12:00 PM	Office Hours John 12PM - 2PM	Lecture Marco, Mike 12:30PM - 1:45PM		Lecture Marco, Mike 12:30PM - 1:45PM	Lab 01 John, Mike, Yana 12PM - 1:45PM
1:00 PM	Office Hours John 12PM - 2PM	Lecture Marco, Mike 12:30PM - 1:45PM	Office Hours John 1PM - 3PM	Lecture Marco, Mike 12:30PM - 1:45PM	Lab 01 John, Mike, Yana 12PM - 1:45PM
2:00 PM	Lab 03 Mike, Yana, Eben 2PM - 3:45PM		Office Hours John 1PM - 3PM		Lab 02 John, Mike, Yana 2PM - 3:45PM
3:00 PM	Lab 03 Mike, Yana 2PM - 3:45PM	Office Hours Eben 3PM - 5PM		Office Hours Eben 3PM - 5PM	Lab 02 John, Mike, Yana 2PM - 3:45PM
4:00 PM		Office Hours Eben 3PM - 5PM		Office Hours Eben 3PM - 5PM	
5:00 PM	Section A John 5PM - 6PM	Section B Christian 5PM - 6PM			

CSC 211?

- Introduction to Programming
  - focus on **problem solving**

Language of choice: **C/C++**. Prior programming experience is not strictly necessary.

- Introduction to Object Oriented Programming
  - classes, objects, inheritance
- Review of elementary CS techniques, algorithms and data structures
  - e.g. recursion, sorting, linked lists

Prerequisites: **CSC 106** or major in Computer Engineering

Tentative Schedule

Week 1	–	
	09/05/2019	Introduction to 211, Computer Systems, Programming Languages
	Lab	Hello 211, IDE Setup, Basic Shell Commands
Week 2	09/10/2019	Introduction to C++ (input/output), Compiler, Linker
	09/12/2019	Variables, Assignments, Basic Data Types, Expressions
	Lab	Algorithms, Problem Design, Pseudo-code Exercises
Week 3	09/17/2019	Further look into DataTypes, Number Systems
	09/19/2019	Boolean Expressions, Simple Branching, Multiway Branching
	Lab	Programming Exercises (branching)
Week 4	09/24/2019	Loops (for, while, do while)
	09/26/2019	Nested Loops (examples)
	Lab	Programming Exercises (loops and basic sorts)
Week 5	10/01/2019	Functions, Scope of Variables, Parameter passing (by value, by reference)
	10/03/2019	Call Stack and Examples
	Lab	Using the Debugger, Programming Exercises (functions)

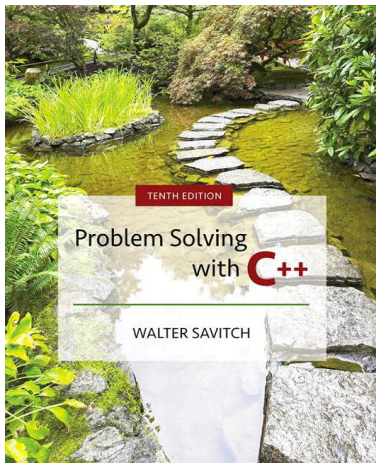
Tentative Schedule

Week 6	10/08/2019	Arrays, Arrays and Functions
	10/10/2019	Multidimensional Arrays
	Lab	Midterm Exam (weeks 1 to 5)
Week 7	10/15/2019	Make up day for Columbus Day
	10/17/2019	Strings (C style and string class)
	Lab	Programming Exercises (strings)
Week 8	10/22/2019	Pointers
	10/24/2019	Allocating Arrays and Multidimensional Arrays
	Lab	Programming Exercises (pointers) and Dynamic Memory Allocation
Week 9	10/29/2019	Recursion and Examples
	10/31/2019	Binary Search and Unimodal Arrays
	Lab	Programming Exercises (tracing recursion, recursion trees)
Week 10	11/05/2019	Advanced Recursion (Backtracking)
	11/07/2019	Structs
	Lab	Advanced Recursive Problems

Tentative Schedule

Week 11	11/12/2019	Classes, Data Members and Methods (Encapsulation)
	11/14/2019	Constructors, Destructors
	Lab	Implementing Classes (source/headers), Arrays and Classes, Copy Constructors
Week 12	11/19/2019	Class Inheritance
	11/21/2019	Friend Functions, Overloading Operators
	Lab	Develop a Class (overloaded operators)
Week 13	11/26/2019	Midterm Exam (weeks 6 to 10)
	11/28/2019	Thanks giving Recess
Week 14	12/03/2019	Command Line Arguments, stdin/stdout, Streams
	12/05/2019	Singly Linked Lists
	Lab	Implementing a Linked List Class and read/write from files (use CLA)
Week 15	12/10/2019	STL Containers
	–	
Finals		Final Exam (cumulative but focus on weeks 11 to 14)

Required textbook



No need to buy

MyLab Programming

URI CAMPUS STORE

THE UNIVERSITY OF RHODE ISLAND

amazon

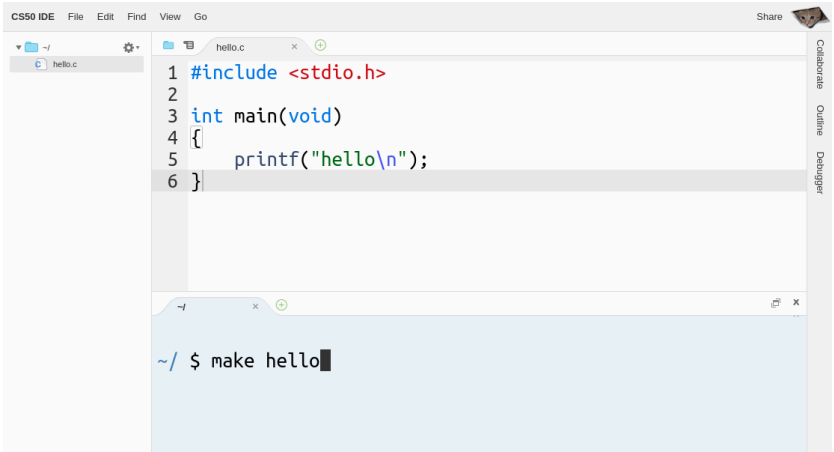
Pearson

C/C++?

Recommended Tools

- although you are free to use **any IDE** on **any platform**, we will grade all assignments using **g++ on a Linux machine**
- CS50 IDE is strongly **recommended** !
- alternatives:
  - vim, g++, gdb (running on Linux)
  - VSCode

CS50 IDE



Grading (subject to change)

- Assignments
  - ~8 programming assignments (25%)
  - lab attendance (5%)
- Exams
  - 2 midterm exams (40%)
  - 1 final exam (30%)

gradescope

All exams are based on textbook chapters and lecture materials

Programming Assignments

- Discussions and collaboration are allowed, however you **must write your own code**
- All programming assignments will be **automatically** graded on Gradescope
  - no less than 7 days to complete
  - late submissions are **NOT** accepted

Plagiarism?

- just **don't do it**
- if you get caught (chances are very high), your name(s) will be immediately reported for further sanctions

Plagiarism

File 1	File 2	Lines Matched
spring-19/submission_15826983/functions.cpp (89%)	spring-19/submission_15857164/functions.cpp (94%)	167
spring-19/submission_15858590/functions.cpp (47%)	spring-19/submission_15860745/functions.cpp (88%)	161
spring-19/submission_15845050/functions.cpp (52%)	spring-19/submission_15859763/functions.cpp (64%)	151
spring-19/submission_15845050/functions.cpp (50%)	spring-19/submission_15854554/functions.cpp (42%)	122
spring-18/submission_6696725/functions.cc (48%)	spring-18/submission_6706969/functions.cc (66%)	91
fall-18/submission_9926606/functions.cpp (72%)	spring-19/submission_15843429/functions.cpp (61%)	128
spring-18/submission_6697832/functions.cc (56%)	spring-18/submission_6706969/functions.cc (60%)	117
spring-18/submission_6696725/functions.cc (44%)	spring-18/submission_6701298/functions.cc (44%)	85
spring-19/submission_15849001/functions.cpp (66%)	spring-19/submission_15856189/functions.cpp (73%)	189
fall-18/submission_9867275/functions.cpp (89%)	spring-19/submission_15860062/functions.cpp (65%)	179
spring-19/submission_15861942/functions.cpp (87%)	spring-19/submission_15863011/functions.cpp (85%)	132
spring-19/submission_15856189/functions.cpp (68%)	spring-19/submission_15857164/functions.cpp (67%)	122
fall-18/submission_9933156/functions.cpp (73%)	spring-19/submission_15857316/functions.cpp (49%)	128
spring-19/submission_15826983/functions.cpp (58%)	spring-19/submission_15856189/functions.cpp (61%)	105
spring-18/submission_6712472/functions.cc (67%)	spring-18/submission_6712960/functions.cc (64%)	149
spring-18/submission_6701298/functions.cc (35%)	spring-18/submission_6706969/functions.cc (48%)	58
spring-18/submission_6696725/functions.cc (35%)	spring-18/submission_6697832/functions.cc (44%)	92
spring-19/submission_15861491/functions.cpp (65%)	spring-19/submission_15861942/functions.cpp (74%)	150
spring-19/submission_15856342/functions.cpp (42%)	spring-19/submission_15858250/functions.cpp (45%)	112
spring-19/submission_15862600/functions.cpp (66%)	spring-19/submission_15863011/functions.cpp (71%)	96
spring-19/submission_15861491/functions.cpp (64%)	spring-19/submission_15863011/functions.cpp (71%)	138
spring-19/submission_15861809/functions.cpp (59%)	spring-19/submission_15862609/functions.cpp (46%)	116
fall-18/submission_9936095/functions.cpp (61%)	spring-18/submission_6700982/functions.cc (42%)	67
spring-19/submission_15861942/functions.cpp (68%)	spring-19/submission_15862600/functions.cpp (62%)	86
spring-19/submission_15849001/functions.cpp (49%)	spring-19/submission_15857164/functions.cpp (54%)	155
spring-19/submission_15857164/functions.cpp (53%)	spring-19/submission_15862216/functions.cpp (53%)	125
spring-19/submission_15862555/functions.cpp (57%)	spring-19/submission_15862609/functions.cpp (44%)	107
fall-18/submission_9856744/functions.cpp (61%)	spring-19/submission_15827542/functions.cpp (47%)	149

How to succeed in this class?

- I do not spend time taking attendance … but …
  - students skipping lectures will (very) likely **fail** this class
- **Organize** your time
  - lectures, labs, discussion sections, programming assignments, exams
- **Participate** and think critically
  - ask questions (lectures, labs, discussion sections, office hours, Piazza)
- Start assignments **early**
  - programming and debugging takes time (especially for all/nothing grading)
  - **avoid** copying/pasting or google'ing answers by all means

How to succeed in this class?

- Skim related textbook section/chapter before coming to lecture
  - read thoroughly afterwards and solve problems/exercises
- Do not expect assignment solutions from the TAs
- Think before you code
  - write pseudocode on paper
- Write code incrementally
  - write, compile, test frequently

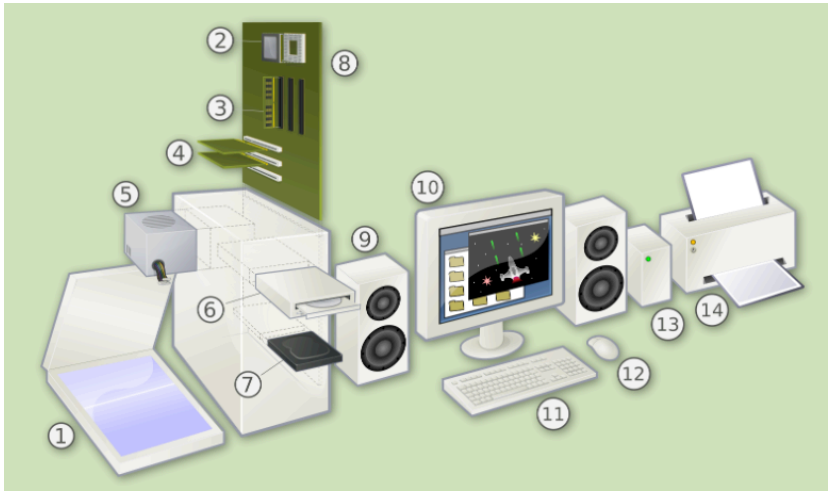
Need help?

- Come to **Office Hours**
- Post questions on **Piazza**
  - answer questions, share information

piazza

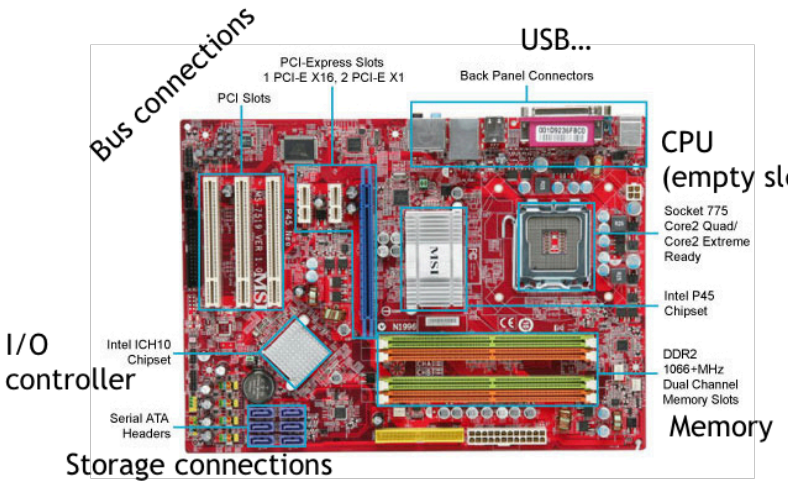
Computer Systems

Computer Components



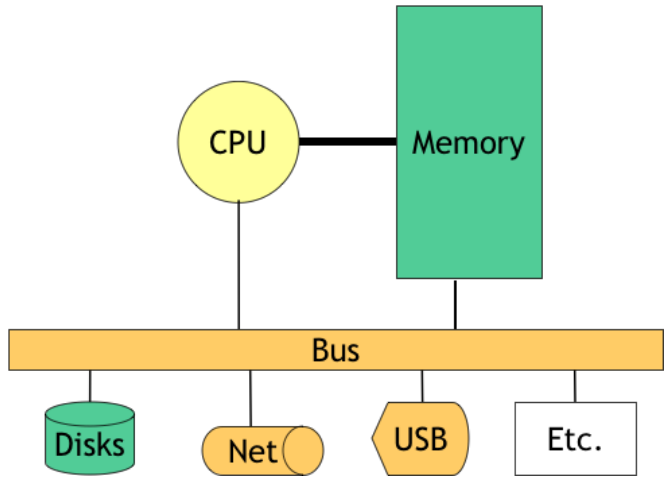
https://bjc.edc.org/bjc-r/cur/programming/6-computers/1-abstraction/07-digital-components.html

Inside a typical computer/laptop



from: CSE 351, University of Washington

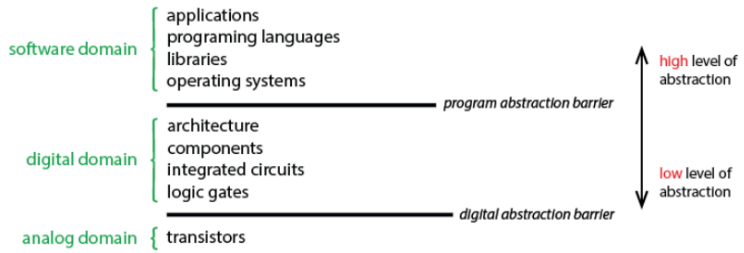
Von Neumann model



from: CSE 351, University of Washington

Abstraction Layers

"the process of removing physical, spatial, or temporal details or attributes in the study of objects or systems in order to focus attention on details of higher importance" [wikipedia]

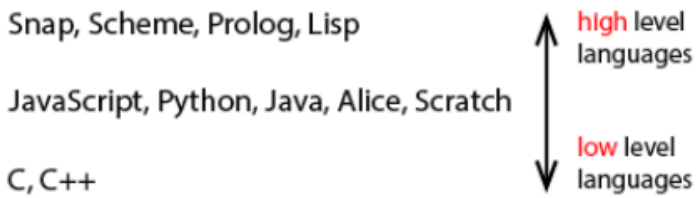


Different people might draw this diagram slightly differently, so don't try to memorize all the levels. The key abstraction levels to remember are software, digital computer hardware, and underlying analog circuit components.

https://bjc.edc.org/bjc-r/cur/programming/6-computers/1-abstraction/01-abstraction.html

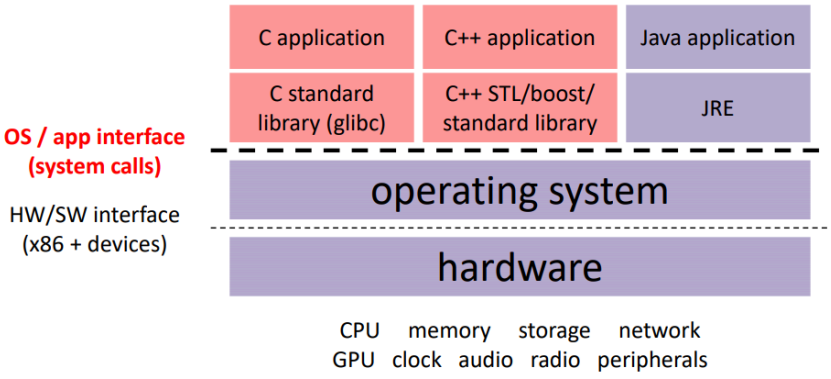
High-Level and Low-Level Languages

A *high-level language* (like Snap*!* or Scheme) includes many built-in abstractions that make it easier to focus on the problem you want to solve rather than on how computer hardware works. A *low-level language* (like C) has fewer abstractions, requiring you to know a lot about your computer's architecture to write a program.



https://bjc.edc.org/bjc-r/cur/programming/6-computers/1-abstraction/03-software-languages.html

Programming applications



from: CSE 333, University of Washington

Compiling C code

```
void sumstore(int x, int y,
              int* dest) {
  *dest = x + y;
}
```

C source file (sumstore.c)

C compiler (gcc -S)

```
sumstore:
        addl    %edi, %esi
        movl    %esi, (%rdx)
        ret
```

Assembly file (sumstore.s)

Assembler (gcc -c or as)

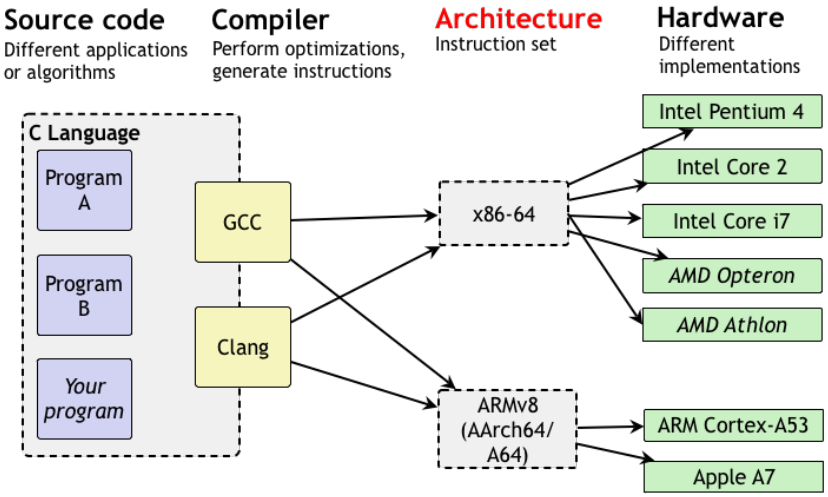
```
400575: 01 fe
        89 32
        c3
```

Machine code (sumstore.o)

C compiler (gcc -c)

from: CSE 333, University of Washington

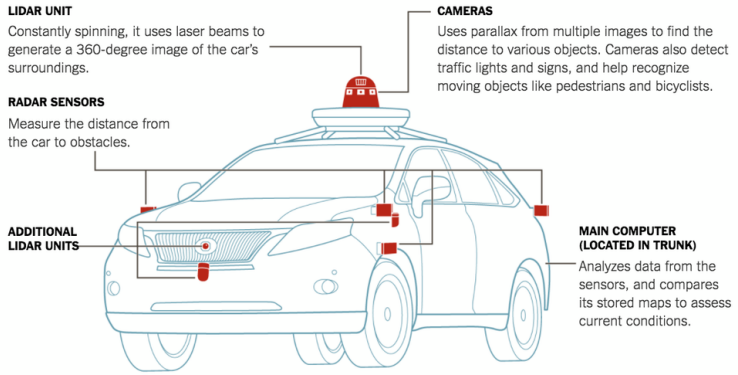
Multiple targets



from: CSE 351, University of Washington

Devices everywhere

Devices everywhere



https://www.nytimes.com/2018/03/19/technology/how-driverless-cars-work.html