Slides by Andrew DeOrio

EECS 280

Programming and Introductory Data Structures

Introduction

Announcements

- Office hours – check the Class Google Calendar
- You should have a CAEN account automatically
  - Google “CAEN Hotline” for help
- Project 1 due in about 1 week
- Please read the syllabus (on CTools)

What you need for EECS 280

- Prerequisites: EECS 182 or EECS 183 or ENGR 101 or ENGR 151

- Basic C++ experience
  - Can you do project 1?
  - Refer to textbook or Internet

What EECS 280 is about

- Computer science is no more about computers than astronomy is about telescopes
  - Often attributed to Edsgar Dijkstra

- What's the point of EECS 280?

What EECS 280 is about

- ... learning *generalizable* CS concepts

What EECS 280 is a

- Cake ... ?
- Did anyone bake a cake this summer? How many eggs did you use?
  - The exact recipe is *not a generalizable* concept!
- What do eggs do in a cake?
  - This is a *generalizable* concept!
- Don’t just memorize a bunch of recipes!

Slide credit: James Juett
Image credit: thecakeblog.com

What EECS 280 is about

- ... learning *generalizable* CS concepts
  - Procedural abstraction
  - Data abstraction
  - Dynamic resource management

What EECS 280 is *not* about

- Learning C++ Programming
- Learning how to write programs that:
  - Calculate statistics
  - Operate on lists and trees
  - Simulate population models
  - Play a game of Euchre
  - Implement an RPN calculator

Some things you will do in EECS 280…

- Learn C++ Programming
- Write programs that:
  - Calculate statistics
  - Operate on lists and trees
  - Simulate population models
  - Play a game of Euchre
  - Implement an RPN calculator

Huh?

- One of the best ways to learn how to program is to write programs
- But in an educational setting the programs aren’t the end goal

- You are!

Slide credit: James Juett

Exercise

- You have 9 balls. Eight have equal weight, one is heavier. Find the heavy ball using only a balance.
- Don't worry about how fast it is

Exercise

- You have 9 balls. Eight have equal weight, one is heavier. Find the heavy ball using only a balance.
- Answer 1: weigh random pairs, check if any resulted in an imbalance
- Is it correct?
  - Yes, because we're trying everything!
- It is efficient? How many weighings, in the worst case?
  - No, because we're trying everything!
  - 8 choose 2 = 28 times

    After 28 tries you know that the last one must be the answer
- Let's make this more efficient

Exercise

- You have 9 balls. Eight have equal weight, one is heavier. Find the heavy ball using only a balance.
- Answer 2: pick one ball and compare it against the other 8, until you find an imbalance
- Is it correct?
  - Yes, because there is exactly one heavy ball
- It is efficient? How many weighings, in the worst case?
  - Maybe ...
  - 7 times, worst case
    After 7 tries, you know that the last ball must be the answer
- Can you make this more efficient?

Exercise

- You have 9 balls. Eight have equal weight, one is heavier. Find the heavy ball using only a balance.
- Answer 3: compare two groups of four balls and reserve one ball. If they are equal, then the reserved ball is the heavy ball. Otherwise, divide the remaining group in two until you find an imbalance.
- Is it correct? What would you do to test this?
  - Test with two groups of equal weight balls – it works!
  - Test with heavy ball in one of the groups – it works!
- It is efficient? How many weighings, in the worst case?
  - It's much better! 3 balances

Exercise

- You have 9 balls. Eight have equal weight, one is heavier. Find the heavy ball using only a balance.
- Can you do it with only two weighings?

How programs run

- We have discussed different algorithms to solve the "heavy ball" problem
- What happens when we code an algorithm using a programming language and run it on a computer?

hello.cpp

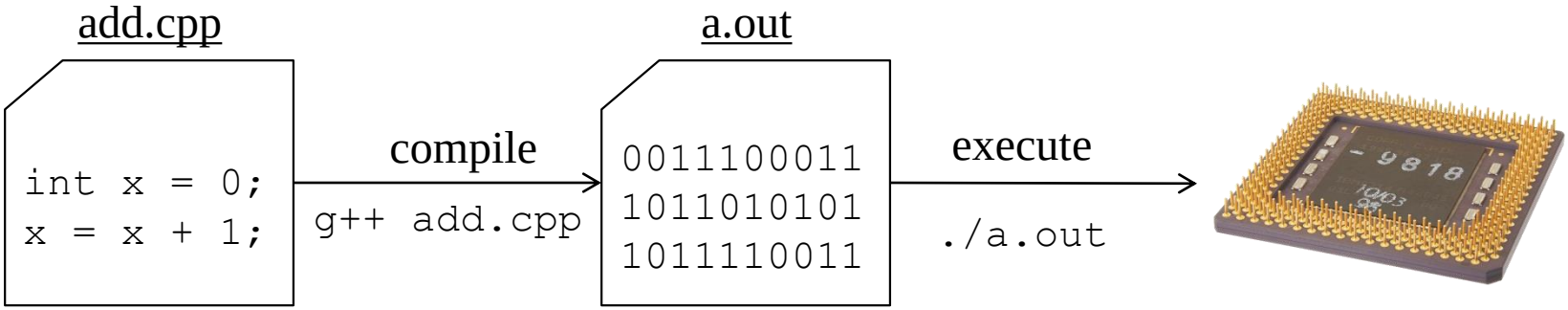
```
#include <iostream>
using namespace std;

int main() {
  cout << "Hello world!"
       << endl;
  return 0;
}
```

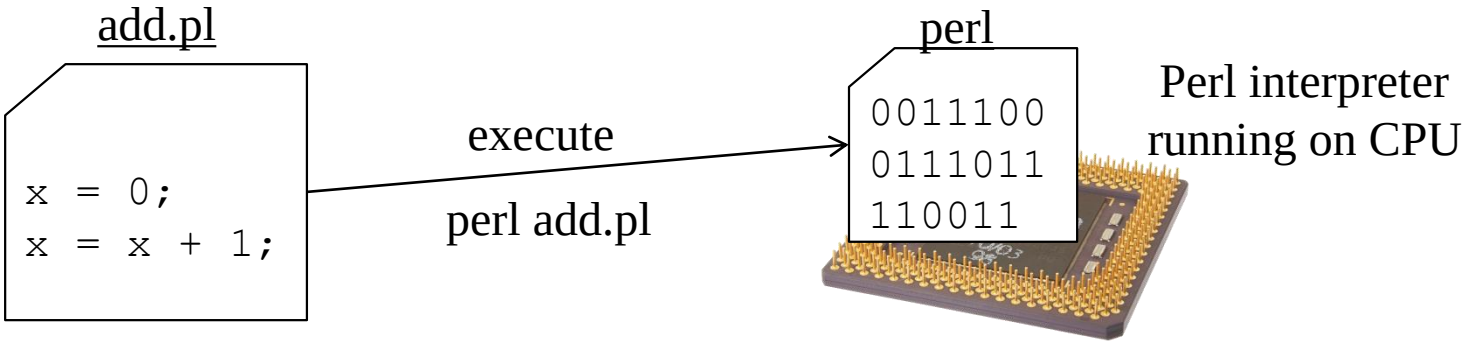
Hello World!

Compiled vs. Interpreted

- Language implementations can be compiled or interpreted
- Compiled: Program is converted into low-level machine code before execution



- Interpreted: Program is run step-by-step during execution



Compiled vs. Interpreted

Compiled

- Faster
  - No execution engine
- Less portable
  - Must recompile
- Less flexible
  - Need to know everything (mostly) at compile-time

Interpreted

- Slower
  - Must go through engine
- More portable
  - Just run!
- More flexible
  - Can change things at runtime

Compilation

- EECS 280 uses compiled C++
  - g++ is our compiler

- Multiple steps inside g++
  1. Preprocessing
  2. Compilation proper
  3. Assembly
  4. Linking

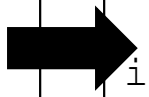
Preprocessing

- Expand `#include` and `#define`, etc.
  - Includes information about `cout` and `endl`
- `g++ -E hello.cpp -o hello.ii`

hello.cpp

```
#include <iostream>
using namespace std;

int main() {
  cout << "Hello world!" << endl;
  return 0;
}
```



hello.ii

```
[ all the iostream info ]

int main() {
  cout << "Hello world!" << endl;
  return 0;
}
```

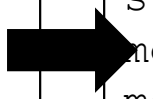
Compilation proper

- Convert C++ code into assembly instructions
  - Complicated, multi-step process
  - Multiple entire courses on this subject, here at Michigan
- Assembly instructions are very close to binary
  - EECS 370
- `g++ -S hello.ii -o hello.s`

hello.ii

```
[ all the iostream stuff ]

int main() {
  cout << "Hello world!" << endl;
  return 0;
}
```

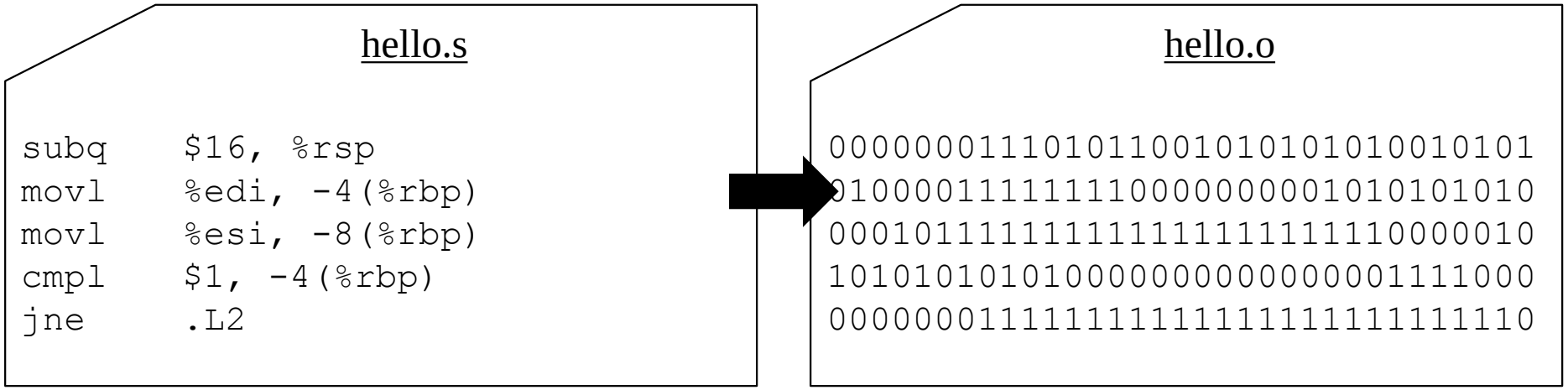


hello.s

```
subq    $16, %rsp
movl    %edi, -4(%rbp)
movl    %esi, -8(%rbp)
cmpl    $1, -4(%rbp)
jne     .L2
```

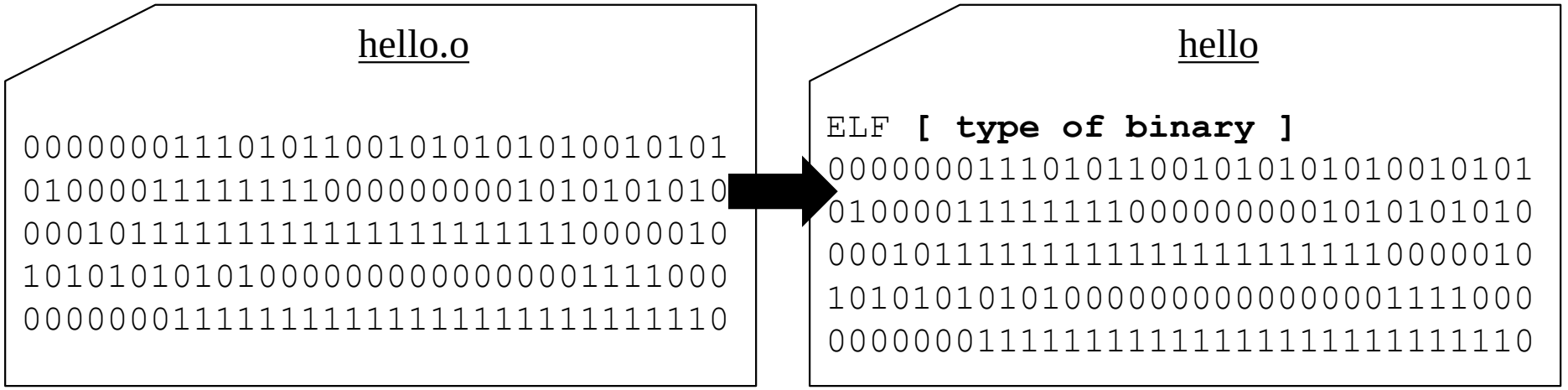
Assembly

- Convert assembly instructions into binary
  - EECS 370
- Result is an **object file**
  - Binary translation of your program
- `g++ -c hello.s –o hello.o`



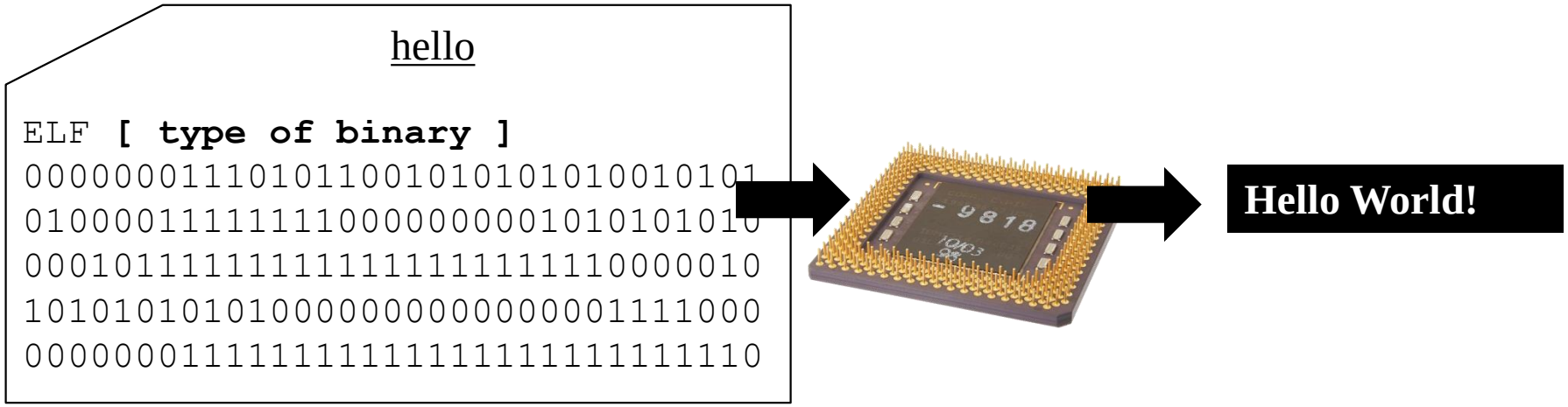
Linking

- Object file doesn't know how to find libraries or other object files
- Linking includes this information in the executable
- `g++ hello.o -o hello`



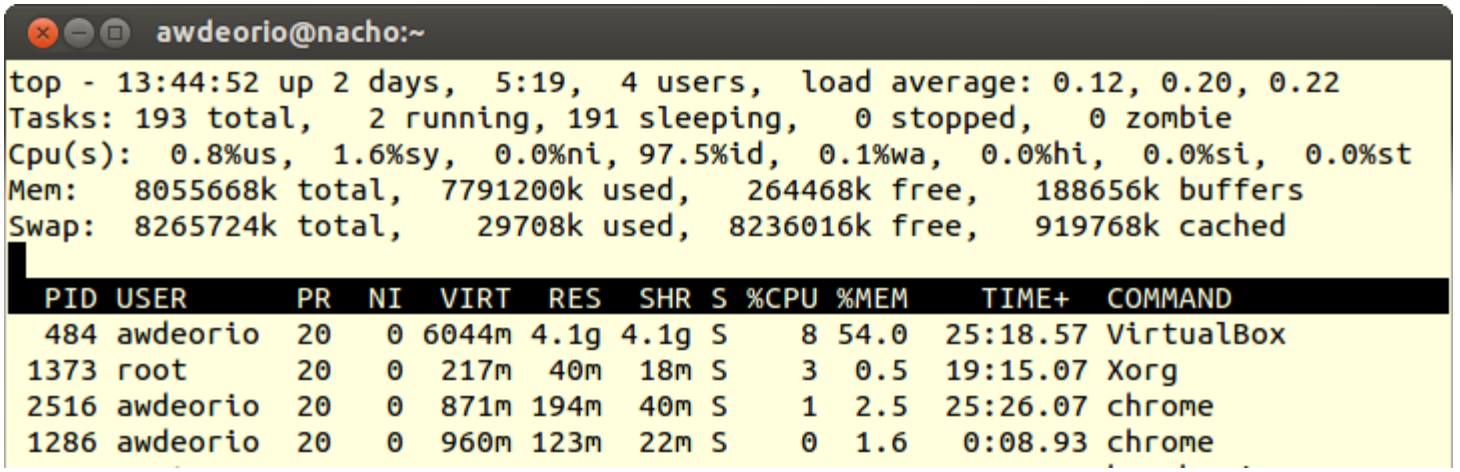
After the Compilation Process

- Result of compilation process is an executable
- Running the executable
  - The shell and the operating system work together
  - Again, a multi-step process
- `./hello`



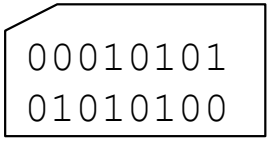
Running the Executable

1. Type program into shell
   - The shell is a program that runs programs
   - The shell provides the command prompt:
2. Shell asks OS to run program
   - OS manages which programs are running and when
   - To see running programs: `top` (type "q" to quit)



Running the Executable

3. OS loads executable into memory
   - Copies “hello” binary from disk to RAM



4. OS begins execution
   - CPU executes binary code

5. Program executes and finishes

3. Control transferred back to OS
   - Cleans up program’s memory (RAM)
   - Shell waits for another command

Putting It All Together

- Hello World Demo

1. Log in
   - Many ways, for example Putty, VNC or SSH
     - See lab 1 for more options
   - One terminal for text editor
   - One terminal for command line
2. Create text file with the program's contents
   - `emacs hello.cpp`
3. Compile
   - `g++ hello.cpp -o hello`
4. Run
   - `./hello`

Diversion!

- Imagine you are shrunk down to the size of a nickel and placed in a blender.
- The blender will turn on in 30 seconds. How do you escape?
- Note: You are not an actual nickel.

Slide credit: James Juett

Lab/Discussion sections

- Discussion sections are lab-style
  - Every discussion will be hands-on coding
  - Bring a laptop
    - Contact course staff if you don't have access to a laptop or tablet

- Due electronically every Sunday
- 5% of the final grade, checked for completion
- Collaboration is encouraged
- Attend any discussion section you want
  - All discussion sections cover the same material

Exams and grades

- The details on exams, projects, and grades can be found in the syllabus

- We will assign grades on a curve, in keeping with past grades given in this course. We will adjust the curve up or down if the class as a whole does better or worse than past instances. In particular, if everyone does exceptionally well, then everyone will get an exceptionally good grade.

Projects

- Projects require:
  - Read and understand a problem specification
  - Design a solution to this problem
  - Implement this solution simply and elegantly
  - Convince yourself of your solution's correctness

- Grading projects will be done by a combination of testing (correctness) and reading (correctness and simplicity/elegance).

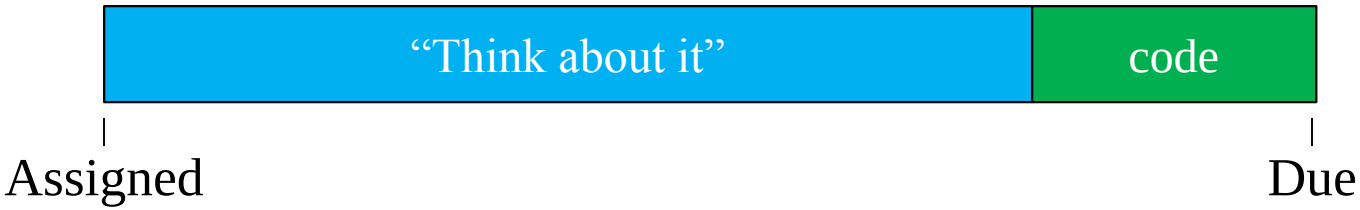
- We will give you a few simple test cases to get started, but we will not tell you everything we will be testing for. It is up to you to figure out your own set of tests, and we will spend a lecture on how to do this.

Projects - autograder

- Submit your projects to https://g280-1.eecs.umich.edu
- The autograder runs several test cases and provides feedback
- Unlimited submissions
- We grade the last submission before the deadline
- Project 1 is open now: submit early and often!

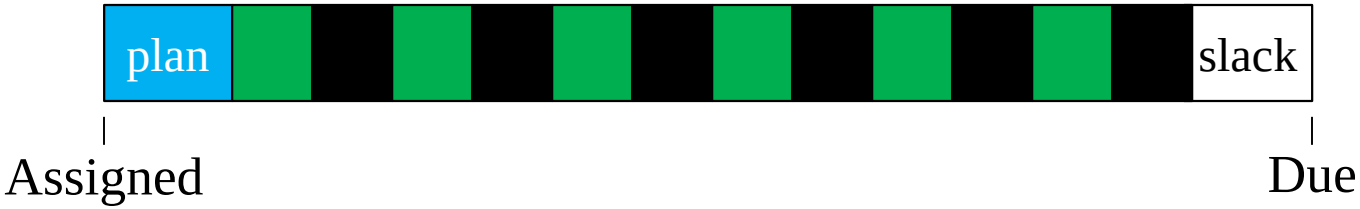
Projects - timeline

- Sample timeline for failure on difficult projects

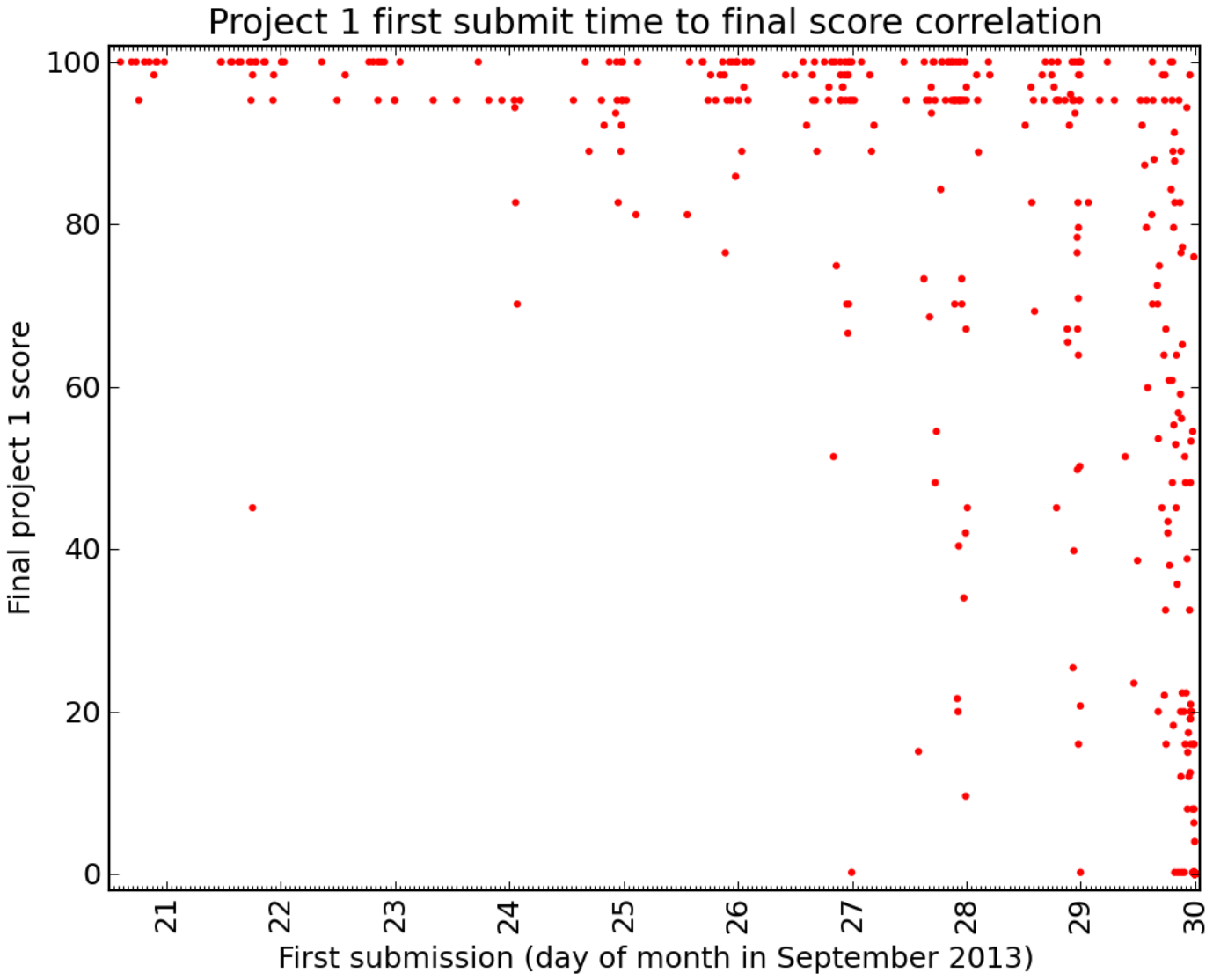


- Sample timeline for success on difficult projects

- Even better



Submission time vs. score



*data from EECS 281, fall 2013

Getting help

- Office hours – see class Google Calendar
- The Piazza Forum
  - If you have a question, and can't get to office hours, feel free to post it here. However, do not post your own code, or "give away" solutions in your question. A member of the course staff will be assigned to the forum during each business day, and it will typically be monitored at other times.
- Please do not send technical questions by email. Either see us in office hours, or post to the forums.

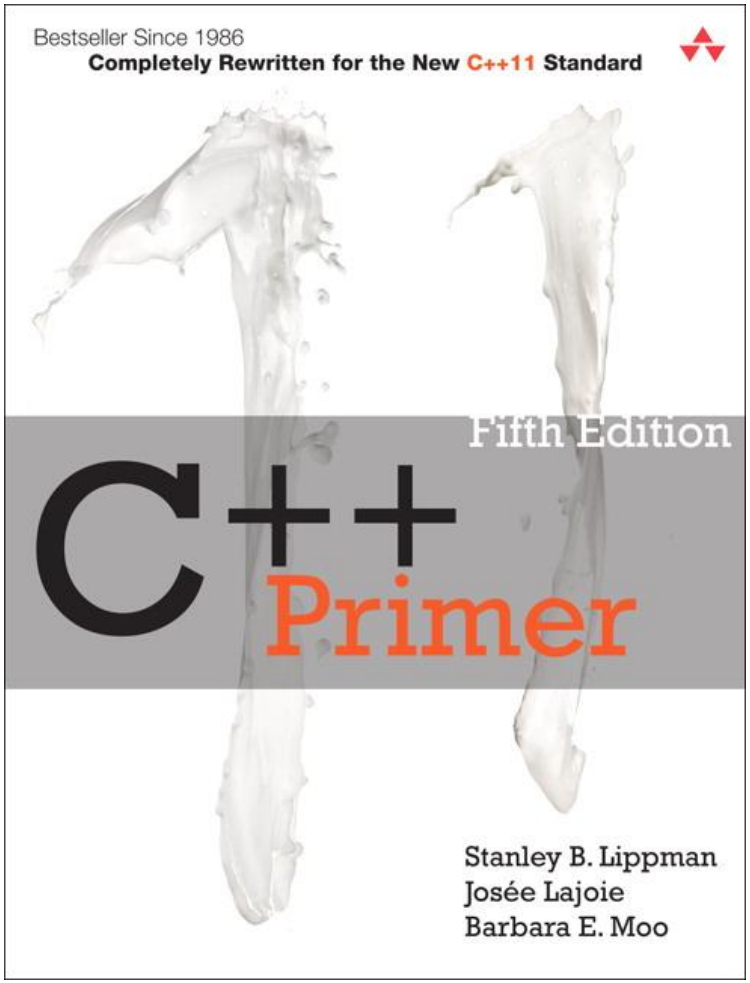
- To contact the staff, please use eecs280staff@umich.edu

Text book

- C++ Primer, by Lippman, Lajoie and Moo. 5th edition

Available in ...

- Print form
- Electronic Kindle edition
- Free electronically through the University Library
  - Number of simultaneous users is limited, and sometimes there may be a wait to access it



Collaboration

- You must complete programming assignment 1 alone
- You may complete programming assignments 2 - 5 *either alone or with a partner*
- All programming assignments in this course are to be done by you or your partnership

Guidelines for Partnerships

- Working in a partnership is optional
- Both you and your partner will still submit your assignments individually, but you should both *write each other's uniqname on the project submission for every file*
- You can cannot change partners in the middle of one project, unless your partner drops the course
- You may change partners only after a project is completed and submitted
- However, you are free to work individually as much as you like or collaborate as much as you like, as long as it is with your partner

Partnership DOs

- Do READ THE SYLLABUS CAREFULLY before programming with another student. You must follow these guidelines, or risk being investigated for an Honor Code Violation.
- Do choose a partner from the current semester of this course.
- Do put both your uniqname and the uniqname of your partner in the comments at the top of all code files. This is important to avoid referral to the honor council.
- Do submit one copy of the project together.

Partnership DON'Ts

- Do not program with someone without understanding these guidelines.
- Do not partner on an assignment with someone who has already solved the problem. Students who do this will not learn as much as those who pair with someone at a similar skill level.
- Do not share code with anyone other than your partner, or a staff member.
- Do not split the work in half. It is important that both partners work on all parts of the program. Splitting the work may harm your or your partner's understanding of that part of the solution.
- Do not partner with anyone who is not currently enrolled in the course.

Academic integrity

- You may not collaborate in any way with people outside your partnership
- See the syllabus on CTools for the full policy
- We use automated and manual cheat-checking
  - Sometimes we don’t have time to check until end of semester
  - Over the past several years, we have reported 2-10% of the class each semester

What you'll get out of EECS 280

- Skills to design, implement, test and debug programs with 1,000+ lines of code
- Prerequisite for future computer science courses
- Credentials for an internship
- Become part of an industry that is changing the world