The Hardware/Software Interface

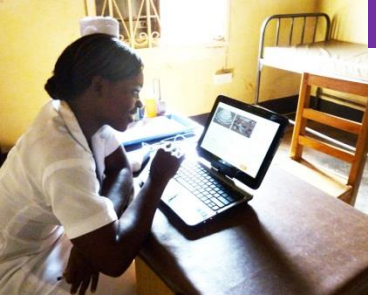
CSE 351 Autumn 2015

Instructor:

Ruth Anderson

Teaching Assistants:

Nicholas Shahan, Eddie Yan, Dylan Johnson, Anthony McIntosh, Cody Ohlsen, Alfian Rizqi, Shan Yang, Aakash Sethi, Frank Sun



Me (Ruth Anderson)

- Grad Student at UW in Programming Languages, Compilers, Parallel Computing
- Taught Computer Science at the University of Virginia for 5 years
- Grad Student at UW: PhD in Educational Technology, Pen Computing
- Current Research: Computing and the Developing World, Computer Science Education
- Recently Taught: data structures, architecture, compilers, programming languages, 142 & 143, data programming in Python, Unix Tools, Designing Technology for Resource-Constrained Environments

The CSE 351 Staff!

Eddie: I'm a first year CSE PhD student. In my former life as an undergrad, I was an avid StarCraft 1/2 fan and player. Some of my favorite pastimes include drinking copious volumes of boba and bench racing. I enjoy cooking in the same way that I enjoy breathing. Feel free to engage me in conversation about how terrible it is to drive in downtown Seattle.

Frank (Chenfan): I'm a junior (second year in the major) originated from Shanghai China. Besides interacting with computers, I enjoy eating like other people enjoy cooking. I love drinking coffee and talking about random stuff. Foosball is also one of my sports. Welcome to drink coffee or play foos with me if we can find a table.

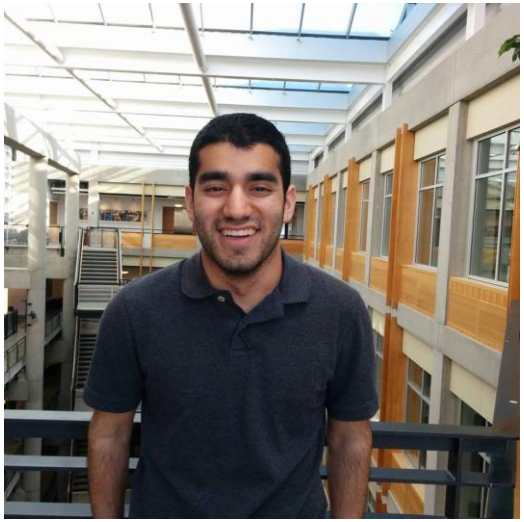
The CSE 351 Staff!

Cody: I am a Computer Engineer undergrad. I am TAing because this was the best class ever! I am a transfer student from community college. I love sports, and if you want me to never stop talking, bring up baseball! I love answering questions, I am very excited to meet new people and I hope everyone here has as much fun as I did in this class.

My name is Alfian Rizqi and I'm a senior. This is my 4th quarter TAing this class. I have interest in sci-fi. I love it when it is snowing and my favorite sport is skiing. If you can guess where I'm from based on my shirt, you're awesome!

The CSE 351 Staff!

Hello! My name is Dylan. I'm a Senior and this is my fourth time TA'ing CSE 351. In my free time I enjoy rock climbing, reading science fiction quadrilogies, astrophotography, and programming video game emulators (after 351 you will be able to also!). Looking forward to a great quarter!



Aakash: Hey everyone! I'm a second-year CSE student. Aside from coding, I love cooking, biking, and listening to podcasts. I'm looking forward to TAing for the first time!

The CSE 351 Staff!

Shan: I am a senior in computer engineering. I went on a road trip to Southern California and Utah in the past 2 weeks. I like playing league of legends, normal not ranked. I have two adorable pet rats.

Hello! I am a Nick, a 5th year masters student from California. I used to work as a theater projectionist so I'm always interested to talk about movies with you. Find me in the lab and let me know if you see a film that you just can't stop thinking about.

The CSE 351 Staff!

Hi! My name is Anthony, I'm a junior from Edmonds WA studying comp sci and this is my first time TAing a class! I also study guitar at UW and in my free time I like listening to records, going to concerts, reading, drinking coffee and hanging out with my two cats. Looking forward to this quarter!

Who are you?

- About 200 registered, likely to be several more
- CSE majors, EE majors, some want-to-be majors
- Please fill out the survey linked on the course web page so we can find out more!
- Introductions on GoPost (see next slide)

Introductions on GoPost

- Name
- Year (1,2,3,4,5)
- Hometown
- Interesting Fact, Hobbies, or what I did over break.

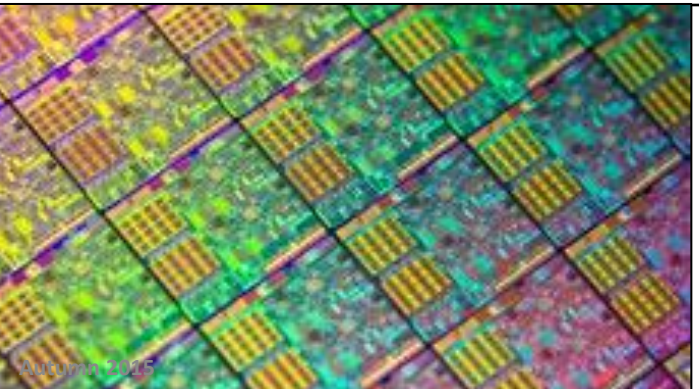
We will email you when we have set this up on GoPost

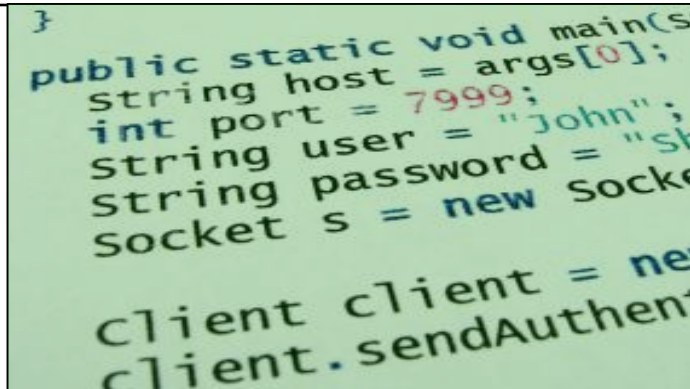
Quick Announcements

- **Website: cse.uw.edu/351**
- **Lab 0, due Monday, 10/5 at 5pm**
  - Make sure you get our virtual machine set up and are able to do work
  - Basic exercises to start getting familiar with C
  - Credit/no-credit
  - Get this done as quickly as possible
- **Section Tomorrow**
  - **Please install the virtual machine BEFORE coming to section**
  - BRING your computer with you to section
  - We will have some in-class activities to help you get started with lab 0

The Hardware/Software Interface

- What is hardware? software?
- What is an interface?
- Why do we need a hardware/software interface?
- Why do we need to understand both sides of this interface?





C/Java, assembly, and machine code

```
if (x != 0) y = (y+z)/x;
```

High Level Language
(e.g. C, Java)

```
      cmpl    $0, -4(%ebp)
      je      .L2
      movl    -12(%ebp), %eax
      movl    -8(%ebp), %edx
      leal    (%edx, %eax), %eax
      movl    %eax, %edx
      sarl    $31, %edx
      idivl   -4(%ebp)
      movl    %eax, -8(%ebp)
.L2:
```

Assembly Language

```
10000011011111000010010000011110000000000
0111010000011000
10001011010001000010010000010100
10001011010001100010010100010100
10001101000001000000010
1000100111000010
11000001111110100011111
11110111011111000010010000011100
10001001010001000010010000011000
```

Machine Code

C/Java, assembly, and machine code

```
if (x != 0) y = (y+z)/x;
```

Compiler

```
    cmpl    $0, -4(%ebp)
    je      .L2
    movl    -12(%ebp), %eax
    movl    -8(%ebp), %edx
    leal    (%edx, %eax), %eax
    movl    %eax, %edx
    sarl    $31, %edx
    idivl   -4(%ebp)
    movl    %eax, -8(%ebp)
.L2:
```

Assembler

```
10000011011111000010010000011110000000000
0111010000011000
10001011010001000010010000010100
10001011010001100010010100010100
10001101000001000000010
1000100111000010
11000001111110100011111
11110111011111000010010000011100
10001001010001000010010000011000
```

High Level Language (e.g. C, Java)

Assembly Language

Machine Code

C/Java, assembly, and machine code

```
if (x != 0) y = (y+z)/x;
```

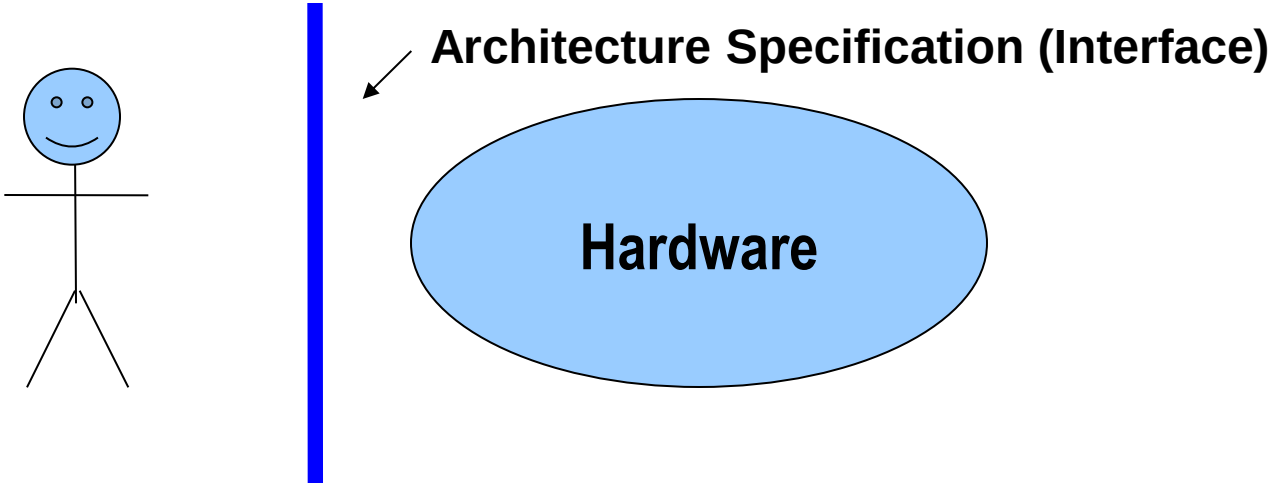
```
      cmpl    $0, -4(%ebp)
      je      .L2
      movl    -12(%ebp), %eax
      movl    -8(%ebp), %edx
      leal    (%edx, %eax), %eax
      movl    %eax, %edx
      sarl    $31, %edx
      idivl   -4(%ebp)
      movl    %eax, -8(%ebp)
.L2:
```

```
10000011011111000010010000011100000000000
0111010000011000
10001011010001000010010000010100
10001011010001100010010100010100
10001101000001000000010
1000100111000010
11000001111110100011111
11110111011111000010010000011100
10001001010001000010010000011000
```

- **The three program fragments are equivalent**
- **You'd rather write C! - a more human-friendly language**
- **The hardware likes bit strings! - everything is voltages**
  - The machine instructions are actually much shorter than the number of bits we would need to represent the characters in the assembly language

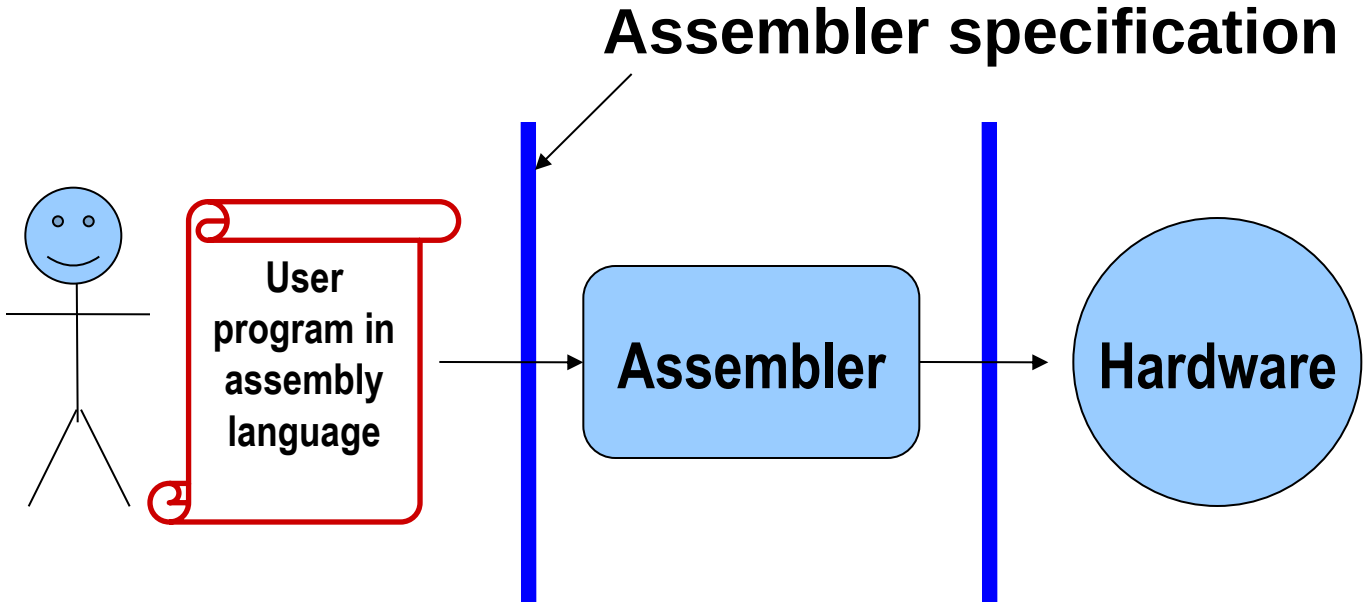
HW/SW Interface: The Historical Perspective

- **Hardware started out quite primitive**
  - Hardware designs were expensive -> instructions had to be very simple – e.g., a single instruction for adding two integers
- **Software was also very basic**
  - Software primitives reflected the hardware pretty closely



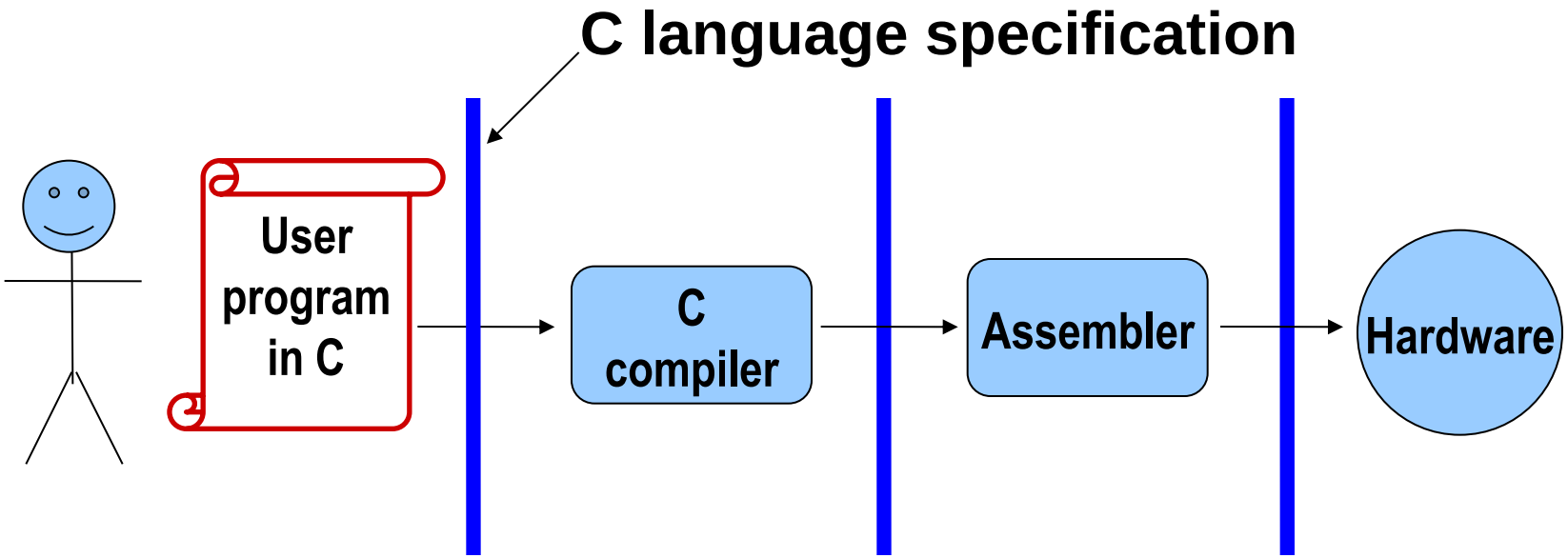
HW/SW Interface: Assemblers

- **Life was made a lot better by assemblers**
  - 1 assembly instruction = 1 machine instruction, but...
  - different syntax: assembly instructions are character strings, not bit strings, a lot easier to read/write by humans
  - can use symbolic names

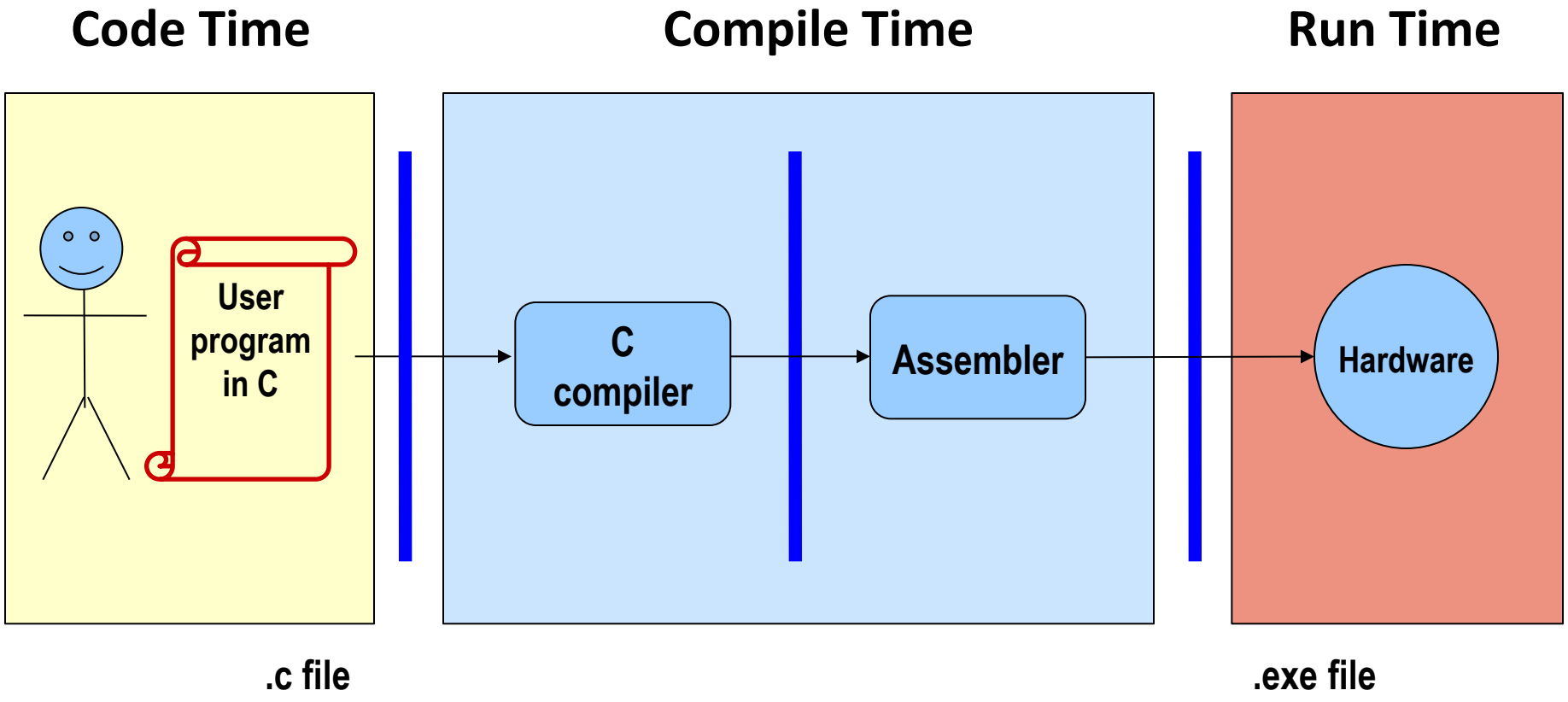


HW/SW Interface: Higher-Level Languages

- **Higher level of abstraction:**
  - 1 line of a high-level language is compiled into many (sometimes very many) lines of assembly language



HW/SW Interface: Code / Compile / Run Times



Note: The compiler and assembler are just programs, developed using this same process.

Outline for today

- Course themes: big and little
- Roadmap of course topics
- How the course fits into the CSE curriculum
- Logistics

The Big Theme: Abstractions and Interfaces

- **Computing is about abstractions**
  - (but we can't forget reality)
- **What are the abstractions that we use?**
- **What do YOU need to know about them?**
  - When do they break down and you have to peek under the hood?
  - What bugs can they cause and how do you find them?
- **How does the hardware (0s and 1s, processor executing instructions) relate to the software (C/Java programs)?**
  - Become a better programmer and begin to understand the important concepts that have evolved in building ever more complex computer systems

Roadmap

C:

```
car *c = malloc(sizeof(car));
c->miles = 100;
c->gals = 17;
float mpg = get_mpg(c);
free(c);
```

Java:

```
Car c = new Car();
c.setMiles(100);
c.setGals(17);
float mpg =
    c.getMPG();
```

Assembly language:

```
get_mpg:
    pushq   %rbp
    movq    %rsp, %rbp
    ...
    popq    %rbp
    ret
```

Machine code:

```
0111010000011000
100011010000010000000010
1000100111000010
11000001111110100011111
```

OS:

Computer system:

Memory & data
Integers & floats
Machine code & C
x86 assembly
Procedures & stacks
Arrays & structs
Memory & caches
Processes
Virtual memory
Memory allocation
Java vs. C

Little Theme 1: Representation

- **All digital systems represent everything as 0s and 1s**
  - The 0 and 1 are really two different voltage ranges in the wires
- **"Everything" includes:**
  - Numbers – integers and floating point
  - Characters – the building blocks of strings
  - Instructions – the directives to the CPU that make up a program
  - Pointers – addresses of data objects stored away in memory
- **These encodings are stored throughout a computer system**
  - In registers, caches, memories, disks, etc.
- **They all need addresses**
  - A way to find them
  - Find a new place to put a new item
  - Reclaim the place in memory when data no longer needed

Little Theme 2: Translation

- **There is a big gap between how we think about programs and data and the 0s and 1s of computers**
- **Need languages to describe what we mean**
- **Languages need to be translated one step at a time**
  - Words, phrases and grammars
- **We know Java as a programming language**
  - Have to work our way down to the 0s and 1s of computers
  - Try not to lose anything in translation!
  - We'll encounter Java byte-codes, C language, assembly language, and machine code (for the X86 family of CPU architectures)

Little Theme 3: Control Flow

- **How do computers orchestrate the many things they are doing?**
- **In one program:**
  - How do we implement if/else, loops, switches?
  - What do we have to keep track of when we call a procedure, and then another, and then another, and so on?
  - How do we know what to do upon “return”?
- **Across programs and operating systems:**
  - Multiple user programs
  - Operating system has to orchestrate them all
    - Each gets a share of computing cycles
    - They may need to share system resources (memory, I/O, disks)
  - Yielding and taking control of the processor
    - Voluntary or “by force”?

Course Outcomes

- **Foundation: basics of high-level programming (Java)**

- **Understanding of some of the abstractions that exist between programs and the hardware they run on, why they exist, and how they build upon each other**
- **Knowledge of some of the details of underlying implementations**
- **Become more effective programmers**
  - Understand some of the many factors that influence program performance
  - More efficient at finding and eliminating bugs
  - Facility with a couple more of the many languages that we use to describe programs and data

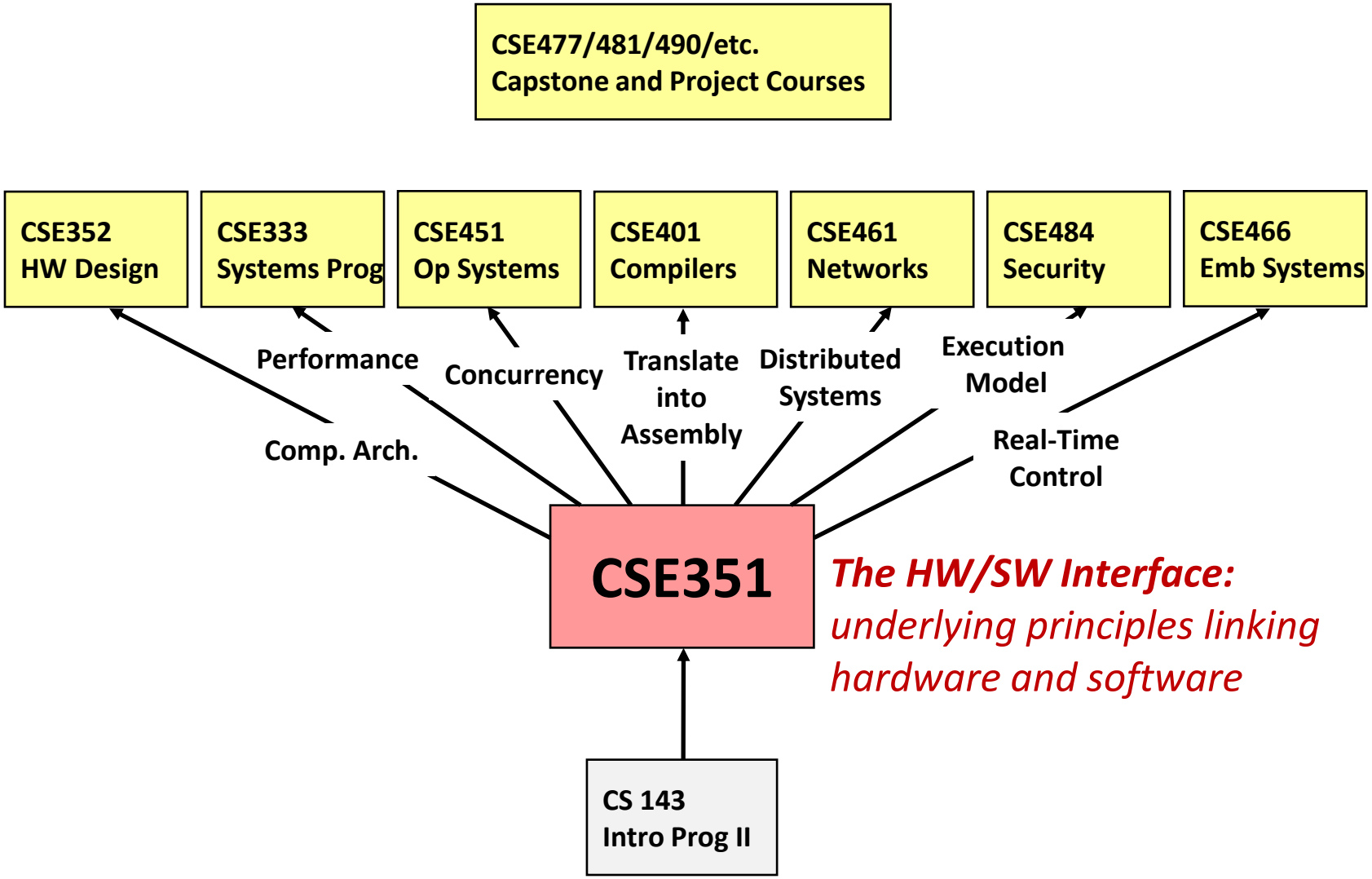
---

- **Prepare for later classes in CSE**

CSE351's role in the CSE Curriculum

- **Pre-requisites**
  - 142 and 143: Intro Programming I and II
  - Also recommended: 390A: System and Software Tools
- **One of 6 core courses**
  - 311: Foundations of Computing I
  - 312: Foundations of Computing II
  - 331: SW Design and Implementation
  - 332: Data Abstractions
  - 351: HW/SW Interface
  - 352: HW Design and Implementation
- **351 provides the context for many follow-on courses**

CSE351's place in the CSE Curriculum

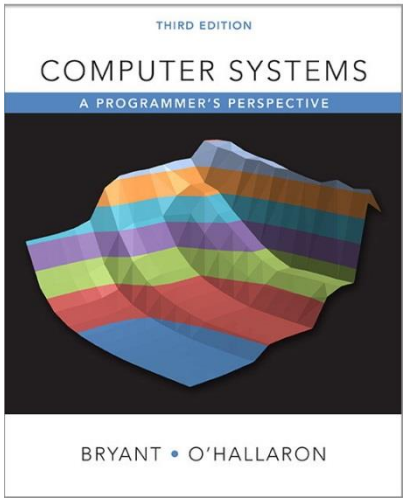


Course Perspective

- **This course will make you a better programmer.**
  - Purpose is to show how software really works
  - By understanding the underlying system, one can be more effective as a programmer.
    - Better debugging
    - Better basis for evaluating performance
    - How multiple activities work in concert (e.g., OS and user programs)
  - Not just a course for hardware enthusiasts!
    - What **every** CSE major needs to know
    - Job interviewers love to ask questions from 351!
  - Provide a context in which to place the other CSE courses you'll take

Textbooks

- **Computer Systems: A Programmer's Perspective, 3rd Edition**
  - Randal E. Bryant and David R. O'Hallaron
  - Prentice-Hall, 2015
  - http://csapp.cs.cmu.edu
  - 3rd edition includes complete rewrite of chapter 3
    - All code examples in x86-64
    - http://csapp.cs.cmu.edu/3e/changes3e.html
  - This book really matters for the course!
    - How to solve labs
    - Practice problems typical of exam problems



- **A good C book – any will do**
  - The C Programming Language (Kernighan and Ritchie)
  - C: A Reference Manual (Harbison and Steele)

Course Components

- **Lectures (29)**
  - Introduce the concepts; supplemented by textbook
- **Sections (10)**
  - Applied concepts, important tools and skills for labs, clarification of lectures, exam review and preparation
- **Written homework assignments (4)**
  - Mostly problems from text to solidify understanding
- **Programming labs/assignments (5, plus "lab 0")**
  - Provide in-depth understanding (via practice) of an aspect of system
- **Exams (midterm + final)**
  - Test your understanding of concepts and principles
  - Midterm is scheduled for Wednesday, November 4, in class
  - Final will be joint with lectures A & B, Wed Dec 16, 12:30-2:20pm in KNE 120

Resources

- **Course web page**
  - cse.uw.edu/351
  - Schedule, policies, labs, homeworks, and everything else
- **Course discussion board**
  - Keep in touch outside of class – help each other
  - Staff will monitor and contribute
- **Course mailing list – check your @uw.edu**
  - Low traffic – mostly announcements; you are already subscribed
- **Office hours, appointments, drop-ins**
  - We will spread our office hours throughout the week
- **Staff e-mail: cse351-staff@cse.uw.edu**
  - For things that are not appropriate for the discussion board
- **Anonymous feedback**
  - Any comments about anything related to the course where you would feel better not attaching your name (we'll provide a response in class)

Policies: Grading

- **Exams (45%): 15% midterm, 30% final**
- **Written assignments (20%): weighted according to effort**
  - We'll try to make these about the same
- **Lab assignments (35%): weighted according to effort**
  - These will likely increase in weight as the quarter progresses
- **Late days:**
  - 3 late days to use as you wish throughout the quarter – see website
- **Collaboration:**
  - http://www.cse.uw.edu/education/courses/cse351/15au/policies.html
  - http://www.cse.uw.edu/students/policies/misconduct

Other details

- **Consider taking CSE 390A Unix Tools, 1 credit, useful skills**
- **Office hours will be held this week, check web page for times**
- **Lab 0, due Monday, 10/5 at 5pm**
  - On the website
  - Install CSE home VM early, make sure it works for you
  - Basic exercises to start getting familiar with C
  - Get this done as quickly as possible
- **Section Tomorrow**
  - **Please install the virtual machine BEFORE coming to section**
  - BRING your computer with you to section
  - We will have some in-class activities to help you get started with lab 0

Welcome to CSE351!

- **Let's have fun**
- **Let's learn – together**
- **Let's communicate**
- **Let's make this a useful class for all of us**

- **Many thanks to the many instructors who have shared their lecture notes – I will be borrowing liberally through the qtr – they deserve all the credit, the errors are all mine**
  - CMU: Randy Bryant, David O'Halloran, Gregory Kesden, Markus Püschel
  - Harvard: Matt Welsh (now at Google-Seattle)
  - UW: Gaetano Borriello, Luis Ceze, Peter Hornyack, Hal Perkins, Ben Wood, John Zahorjan, Katelin Bailey