The Hardware/Software Interface

CSE351 Winter 2014

Instructor:

Mark Oskin

Teaching Assistants:

Matthew Dorsett, Benjamin Du, Whitney Schmidt, Mark Wyse

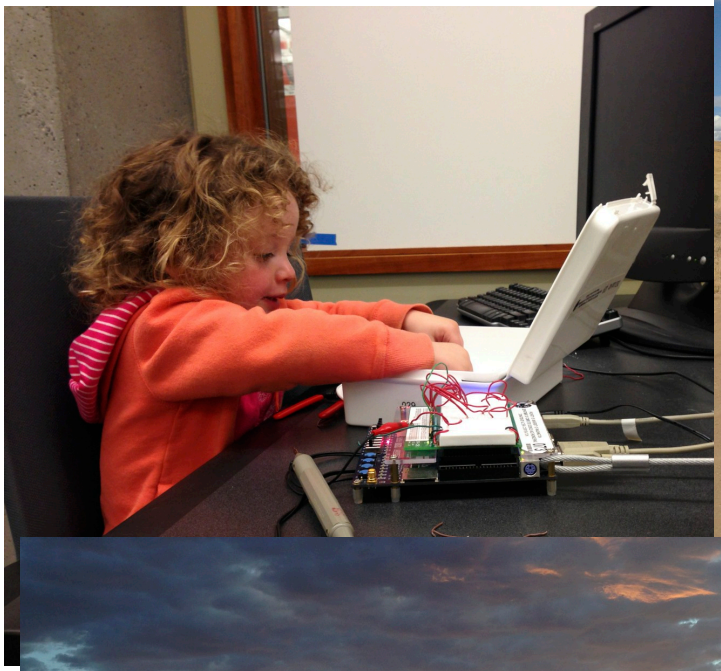
FAQ

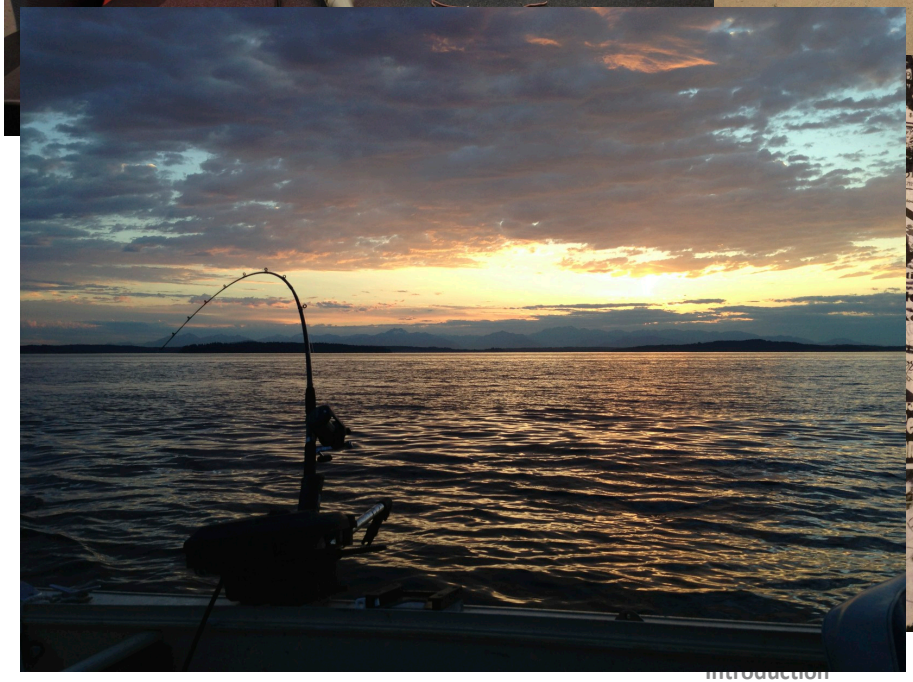
- If you are not officially enrolled, do not worry

Who are you?

- More or less 60
- 20% non-major, 80 major
- 20% CE / 80 CS
- Chicago, India, China, Mexico, Israel, DC

Mark Oskin

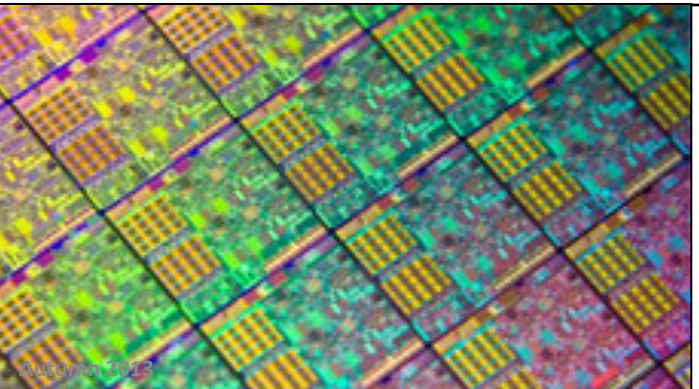


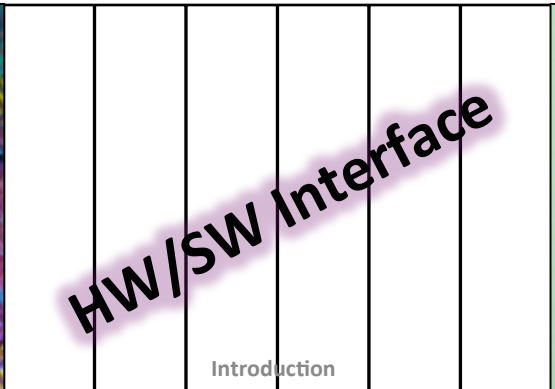


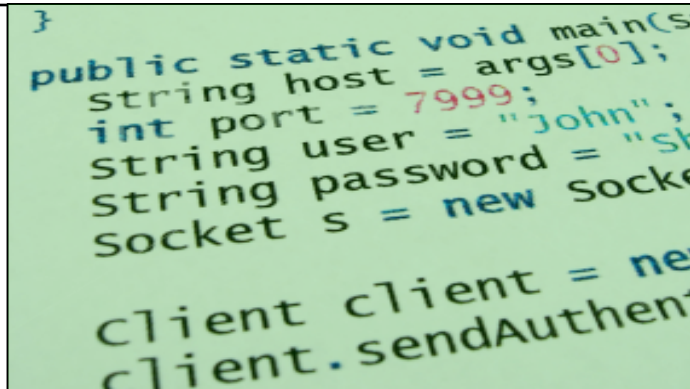

Happy Hour
2 Dr. Saloon March 26
w/Spiderface & Piston Ready
Vikings For A Cause
UBURPER
SKATEBOARDS

The Hardware/Software Interface

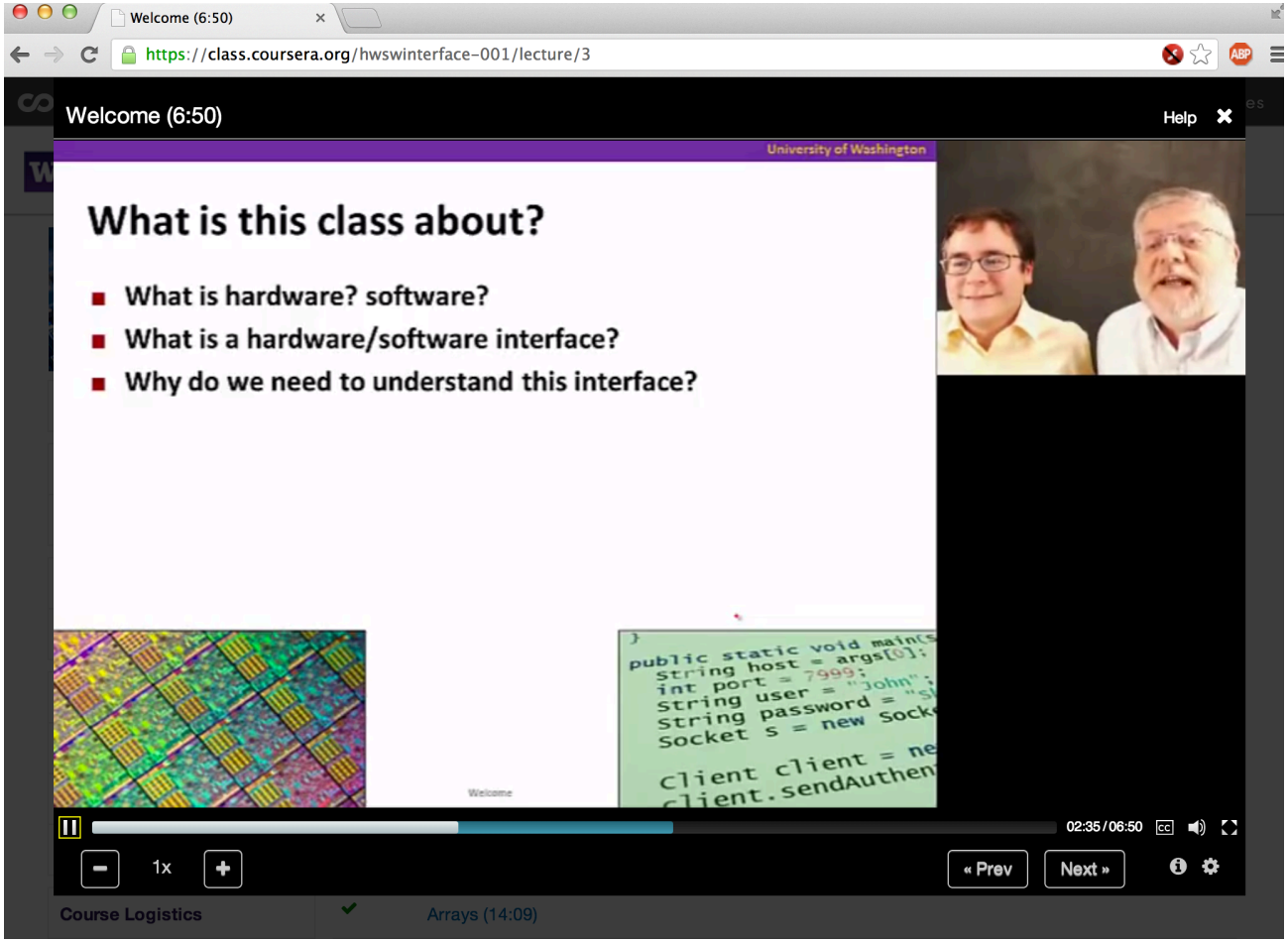
- What is hardware? software?
- What is an interface?
- Why do we need a hardware/software interface?
- Why do we need to understand both sides of this interface?







A new era: this class is all online (sort-a)



IDEA: Intro / Discuss / Explore Advanced

- **More or less on a weekly basis we'll cycle through a set of types of days**
- **Introduction**
  - Intro / Overview / precise roadmap of what to watch
- **Discussion**
  - By this time you've reviewed the on-line lectures and you we'll talk about the concepts and assist you with the homework and labs
- **Exploration & Advanced topics**
  - We'll venture off into advanced topics on the same material covered that week. Possibly some additional reading material will be required.

With great power comes great responsibility…

- **By moving to this hybrid lecture format, you will get the following benefits:**
  - You get to learn more! Seriously, you are here to learn right? You're paying for this education, so lets see how far we can go together!
  - You get to learn the basics at a pace better suited for you and at a time and place better suited for you.
  - Ostensibly you can miss a 'D' day and it won't harm your ability to follow along in class one bit. You could also miss an 'I' day but my goal with 'I' days is to make the time so super useful for you that you find it saves you time to come to class.
- **But you will hate this class – fast – if you don't do your part**
  - *Actually* watch the videos, and preferably before the D day lecture.
  - Come to the EA, yes it's part of the class and you *are* tested on it

Quick Announcements

- **Website: cs.uw.edu/351**
- **Section information to be updated soon…**
- **Lab 0 Will be released today, due next Monday**
  - Basic exercises to start getting familiar with C
  - Credit/no-credit
- **NO LECTURE 1/13 – I'm in Washington DC**

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
11000001111110100001111
11110111011111000010010000011100
10001001010001000010010000011000
```

C/Java, assembly, and machine code

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

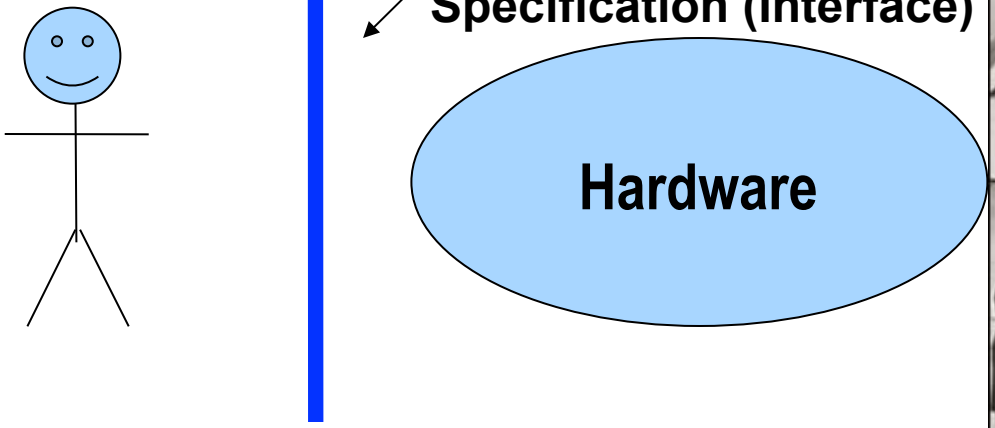
```
    cmpl    $0, -4(%ebp)
    je      .L2
    movl    -12(%ebp), %eax
    movl    -8(%ebp), %edx
    leal    (%edx, %eax), %eax
    movl    %eax, %edx
    sarl    $31, %edx
    idivl   -4(%ebp)
    movl    %eax, -8(%ebp)
.L2:
```

```
10000011011111000010010000011100000000000
0111010000011000
10001011010001000010010000010100
10001011010001100010010100010100
10001101000001000000010
1000100111000010
11000001111110100001111
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

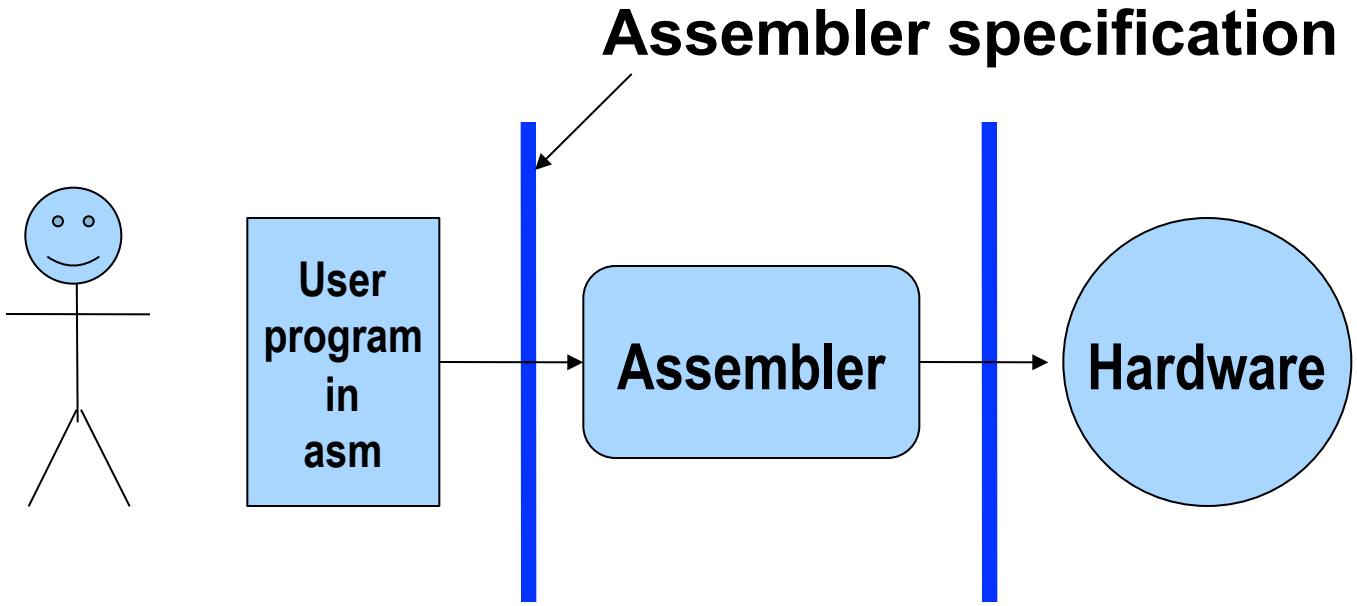
HW/SW Interface: The Historical Perspective

- **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



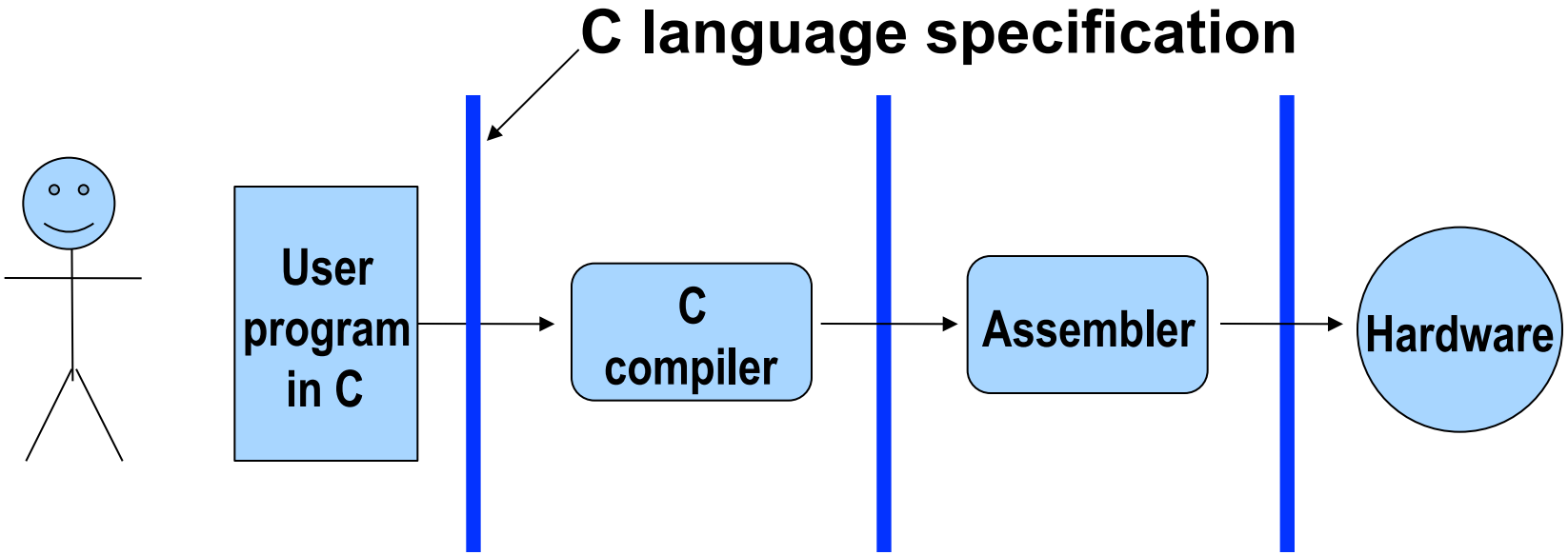
HW/SW Interface: Assemblers

- **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

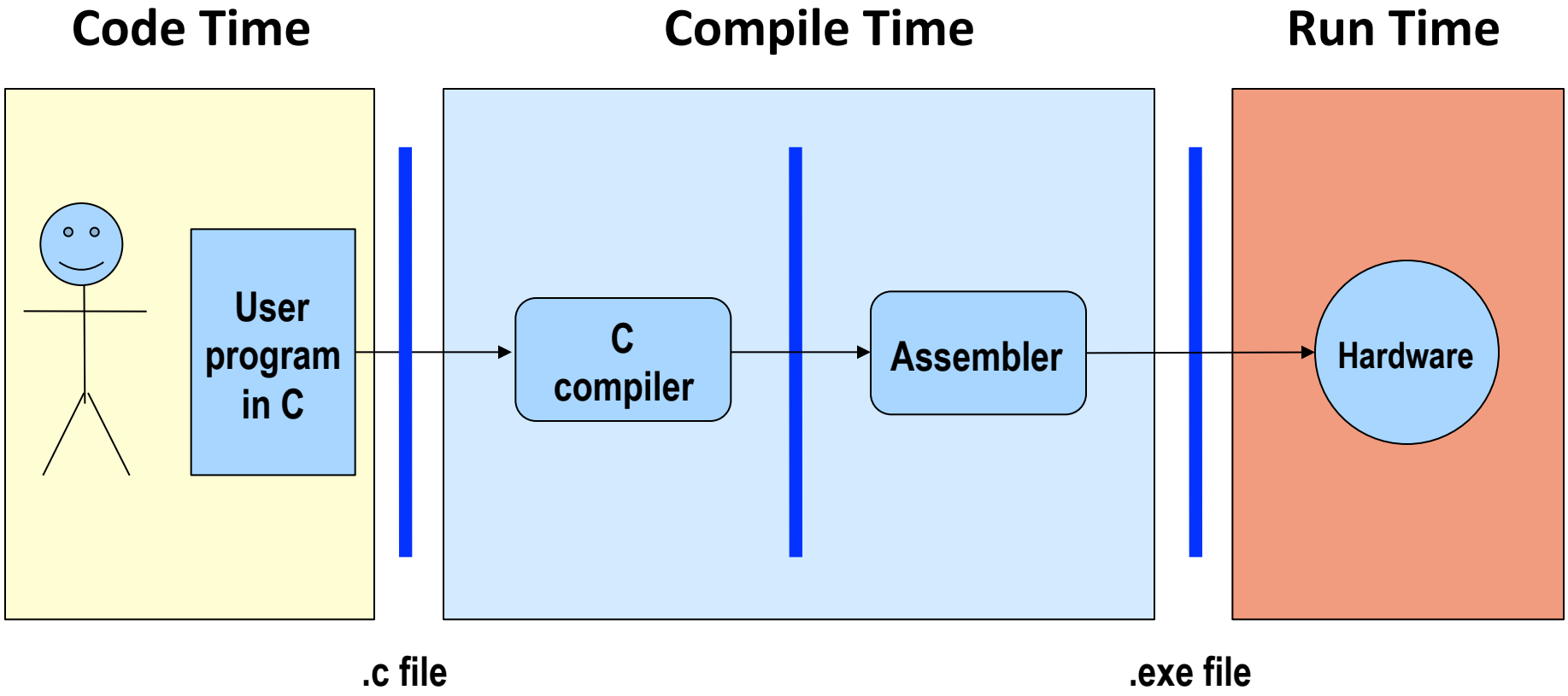


HW/SW Interface: Higher-Level Languages

- **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
- Three important realities
- How the course fits into the CSE curriculum
- Logistics

The Big Theme: Interfaces and Abstractions

- **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:

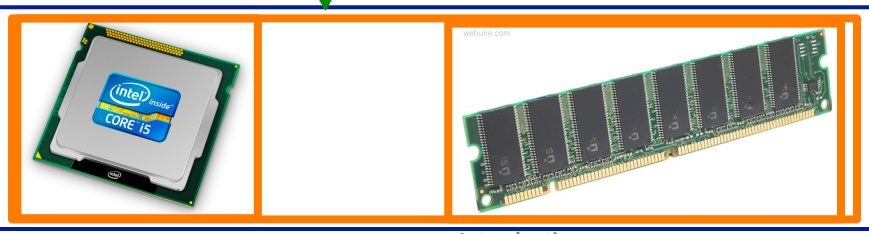
```
get_mpg:
    pushq   %rbp
    movq    %rsp, %rbp
    ...
    popq    %rbp
    ret
```

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"?

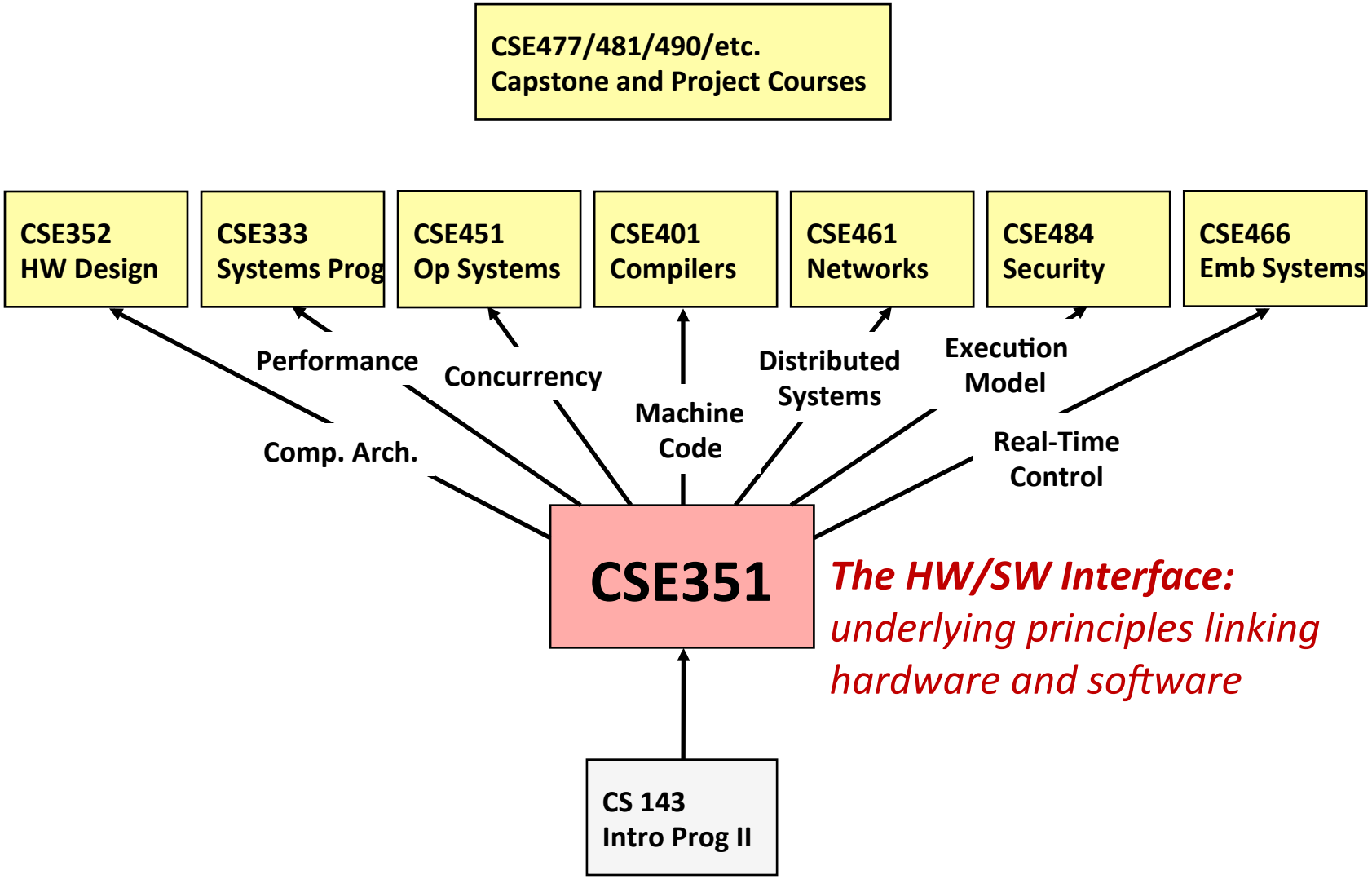
Course Outcomes

- **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**
  - More efficient at finding and eliminating bugs
  - Understand some of the many factors that influence program performance
  - 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



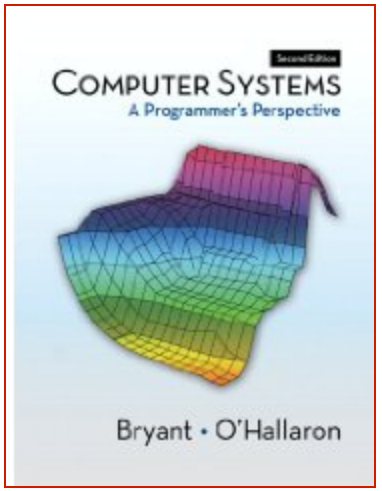
The HW/SW Interface:
underlying principles linking hardware and software

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 dedicated hackers
    - 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, 2nd Edition**
  - Randal E. Bryant and David R. O'Hallaron
  - Prentice-Hall, 2010
  - http://csapp.cs.cmu.edu
  - 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

- **Online lectures**
  - Course videos (courtesy of Gaetano and Luis)
  - Written homework assignments (4)
  - Mostly problems from text to solidify understanding
- **Labs (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
  - Final exam is scheduled for March 19th 2:30 – 4:20 here!

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 – mine are by appt. Txt 206-293-9456**
- **Staff e-mail: cse351-staff@cse.uw.edu**
  - Things that are not appropriate for discussion board or better offline
- **Anonymous feedback**
  - Any comments about anything related to the course where you would feel better not attaching your name; *however,* I really don't take anything personally and much prefer non-anonymous feedback because then we can have a dialogue on how to make things better, not just a datapoint that says something is wrong.

Policies: Grading

- **Exams (40%): 15% midterm, 25% final**
- **Written assignments (20%): weighted according to effort**
  - We'll try to make these about the same
- **Lab assignments (40%): weighted according to effort**
  - These will likely increase in weight as the quarter progresses
- **Late days:**
  - You can turn in *anything* 2 days late *if* you write an excuse, formatted as a Haiku and *email* it *on or before* the original due date. This applies to anything due *before* the final day of class.
- **Collaboration:**
  - http://www.cse.uw.edu/education/courses/cse351/13au/policies.html
  - http://www.cse.uw.edu/students/policies/misconduct

Other details

- **Consider taking CSE 390A, 1 credit, useful skills**
- **Lab 0 due Monday 5pm.**
  - On the website
  - Non-majors: should be able to work without CSE accounts, but …
  - Install CSE home VM
  - Thursday section on C and tools

Near term

- Today: Class overview
- Wednesday: (I day): Memory, data and addressing & numbers
- Friday: (D day): Discussion/HW/Lab assistance
- Monday: -- Canceled – I'm in Washington DC working for d'man
- Wednesday: (EA day) – history of memory / data + GPU + technology changes in memory/architecture

Videos to watch

- **Memory, Data, addressing**
  - 1-6 (takes about an hour)
- **Number representation**
  - 7-14 (takes about an hour)

- **http://courses.cs.washington.edu/courses/cse351/13au/videos.html**

Welcome to CSE351 – Hybrid Ediction!

- Let's have fun – there's time to
- Let's learn – together
- Let's communicate – don't be a deer in the headlights!
- Let's make our in-class time *more* productive through hybrid education, not less.

- Many thanks to Gaetano and Luis for spending an enormous amount of time last year recording themselves.