We made it! 😊

C:

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

Java:

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

Assembly language:

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

Machine code:

```
0111010000011000
100011010000010000000010
1000100111000010
110000011111101000011111
```

Computer system:

```
Windows 8
Mac
```

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

Winter 2016
Course Wrap-Up
Today

- Last few pieces of how the Java VM works
  - Bytecode format

- Victory lap and high-level concepts
  - More useful for “5 years from now” than “next week’s final”

- Imploring you to do your course evaluations, please!

- “Ask me anything”
Final Exam

- Wednesday 2:30PM

- Of course it will be difficult
  - But probably less time pressure than midterm

- Very heavily focused on post-midterm material
Course Evaluations

- Per my email, worth 1 HW extra-credit point

- Really matters, and 90-100% response rate makes them much more useful than 60%
  - Have to guess what sampling bias is for “missing 40%”

- None of you have been professors, so you under-estimate how much we take them into account
  - My first time in 351 and I have my own ideas on what to improve – am I right? Is it “nearly perfect already”?
Victory Lap

A victory lap is an extra trip around the track

- By the exhausted victors (that’s us) 😊

Review course goals

- Slides from Lecture 1
- What makes CSE351 special
First...

Thanks to your awesome TAs!
- Everything has been *crazy* smooth all quarter
Next 7 slides copied without change from Lecture 1

They should make much more sense now!
Welcome!

10 weeks to see the key abstractions “under the hood” to describe “what really happens” when a program runs

- How is it that “everything is 1s and 0s”?
- Where does all the data get stored and how do you find it?
- How can more than one program run at once?
- What happens to a Java or C program before the hardware processor can execute it?
- Why is recursion not even slightly magical?
- And much, much, much more...

An introduction that will:

- Profoundly change/augment your view of computers and programs
- Connect your source code down to the hardware
C/Java, assembly, and machine code

- 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

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

```assembly
cmpl $0, -4(%ebp)
jena.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:
```

```
100000110111110000100100001110000000000 0111010000011000 10001011010001000010010000010100 10001011010001100010010100010100 100011010000010000000010 1000100111000010 110000011111101000001111 111101110111110000100100000011100 10001001010001000010010000011000 0111010000011000 10001011010001000010010000010100 10001011010001100010010100010100 100011010000010000000010 1000100111000010 110000011111101000001111 111101110111110000100100000011100 10001001010001000010010000011000
```
HTTP://XKCD.COM/676/

AN x64 processor is screaming along at billions of cycles per second to run the XNU kernel, which is frantically working through all the POSIX-specified abstraction to create the Darwin system underlying OS X, which in turn is straining itself to run Firefox and its Gecko renderer, which creates a Flash object which renders dozens of video frames every second because I wanted to see a cat jump into a box and fall over.

I am a god.
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
  - Or magnetic positions on a disc, or hole depths on a DVD, or…

- “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
- 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)
    - Not in that order, but will all connect by the last lecture!!!
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 Perspective

■ **CSE351 will make you a better programmer**
  ▪ Purpose is to show how software really works
  ▪ Understanding the underlying system makes you more effective
    ▪ 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 (plus many more details)
    ▪ Job interviewers love to ask questions from 351!
  ▪ Like other 300-level courses, “stuff everybody learns and uses and forgets not knowing”

■ **CSE351 presents a world-view that will empower you**
  ▪ The intellectual tools and software tools to understand the trillions+ of 1s and 0s that are “flying around” when your program runs
Now for fun
Which of the following didn’t Dan say? 😊

- It’s like sending a mule to Auburn to pick up some paper
- It’s like hypnotizing you and moving you out of your chair
- It’s like a polka-dot whoopie cushion
- It’s like smoking one cigarette in your life
- It’s like a 3-person marriage
There’s actually a point here...

- The design of digital computers and the abstractions above them *is* system design
  - *Silly* analogies aside, analogies often do hold...
  - ... engineering is design under constraint, whether its bits or girders
  - ... and is done by humans who leverage their experience

- The humans who over decades designed “standard” computing systems:
  - Have crystallized many amazing concepts: caches, translation tables, jump tables, twos-complement, garbage collection, ...
  - Are no smarter than you 😊
Time permitting...

What else do you want to know?

- About how computers work?
- About life in CSE or UW?
- Cute pictures of my kids? 😊
- ...

Thanks for a great quarter!

- Don’t be a stranger!
  - I love hearing from seniors, alumni, etc.