Suppose you want to run lots of Python programs—all the cool kids are doing it, but unfortunately you have a computer that can only execute x86 assembly. So you decide to write a compiler from Python to x86 assembly.

Furthermore, suppose that you think ML is the best language for writing a compiler; you find the language has particularly effective constructs for implementing compilers. Unfortunately, your computer only has a C to x86 compiler. You know that you can write an awesome compiler in ML that produces efficient x86 code, but you’re not so sure you can write an efficient compiler in C. What do you do?
Getting to Know You: “I, ..., wonder ...”
What is compilation?

Miles: translator between languages
Sam: higher-level to a lower-level
  (easy for humans)
  (easy for computers/machines)

Luke: limited instructions

compiler: Program \textit{high} \rightarrow Program \textit{low}
  (assembly)
What is an interpreter?

A "program" (or machine) that runs other programs

\[ \text{interpreter} : \text{Program} \times \text{Data} \rightarrow \text{Data} \]

\[ \text{interpreter} (\text{some input}, \text{program}, \text{in}) \rightarrow \text{program output} \]

run

give this some other interpreter
Uh oh ...
Some terminology

• **A programming language** is
  
  "computable" — express computation

  "means" — notation to

• **A program** is

  some instance of a programming language

  "sequence of instructions"
An analogy: A cooking recipe

- program | recipe
- interpreter | cook
- execution | actions (e.g. baking)
Syntax and semantics: An analogy

What can be written in the PL

"Grammar"

A program's meaning

(what it does, how it should execute)
Meta Language and Object Language

Spanish

English
Interpreter : Program + Data → Data

"Python Interpreter"

implemented in C

Python (obj)

C (meta)
Compiler : Program $\rightarrow$ Program

$\text{Meta Lang}$ $\rightarrow$ Target Obj Lang

Same Obj Lang
Back to the puzzle ...

• What do we have?

\[
\begin{align*}
\text{C} & \quad \text{T} & \quad \text{x86} \\
x86 & \quad & \quad \text{x86} \\
\quad & \quad & \quad \text{I} \\
\quad & \quad & \quad \text{run}
\end{align*}
\]
Step 0

\[ ML \frac{0}{1} \times 86 \]  
\[ C \]  
\[ \text{(ok code)} \]

\[ ML \frac{0}{1} \times 86 \]  
\[ C \]

\[ ML \frac{1}{1} \times 86 \]  
\[ \text{(ok code)} \]

\[ C \]  
\[ \frac{1}{1} \times 86 \]  
\[ \times 86 \]  
\[ C \]  
\[ \times 86 \]
Step 1

ML \frac{2}{1} x86 awesome

ML

ML \frac{3}{1} x86 awesome

\frac{1}{T} x86

\frac{4}{1} x86 A awesome!

\frac{3}{1} x86 B awesome!

Bootstrapping
Step 2
Step 3
Step 4
Step 5
Compiler Construction

Evan Chang

Meeting 1: Welcome
CSCI 4555/5525, ECEN 4553/5523, Spring 2015
http://www.cs.colorado.edu/~bec/courses/csci4555-s15/
Distraction-Free Classroom

• Let’s turn off our cell phones and wi-fi

... just imagine that we have class at 30,000 feet
Distraction-Free Classroom

• Laptop users, please sit in the back rows, on the sides
Introductions: Your guide this semester

- Office Hours: T 10:45-11:45, R 8:30-9:30, or when the door is open in ECOT 621 and via video conferencing (see moodle)
Introductions

• Who am I?
• About you?
  - What do you want to get out of this class?
Post on Piazza

• Background
  - Comfort with Python?
  - Comfort with x86 Assembly?
  - Experience with building language tools (interpreters, translators)?

• What do you want out of this class?

• Can be private note to me
Introductions

• Introduce yourself to someone you haven’t met before. Two minutes!
Focusing on guiding towards understanding ...

- Project-based course: Build a Python compiler
  - Incrementally in 6 projects (5 two-week projects and 1 one-week project)
  - Two weeks of discussion towards completing the lab, driven by you (in class + on Piazza)!
  - Labs due Friday Saturday
  - No late labs but generous “redo” policy
Traditional Format

Why are we learning this?

How is this relevant?

Our Class

So that’s how you solve question 3!
Discussion, discussion, discussion

• Discussion, not lecture
  - Only meeting I will use slides

• **Please interrupt** at any time!

• It’s completely ok to say:
  - I don’t understand. Please say it another way.
  - Slow down!

• **Course is project-based**
  - Lab assignments prompt the discussion
  - Expectation on you to be active
Oath
Administrivia

• Website
  http://www.cs.colorado.edu/~bec/courses/csci4555-s15/
  - notes, resources, etc.
• Discussion Group: Piazza
• Grades, Feedback, Private Materials: Moodle
• Grader: TBD
Today

• Goals for this course
• Requirements and grading
• Course summary

• Convince you that this course, Compiler Construction, is useful
Goals
We will build compilers. Why?

So we don’t write in assembly

to provide useful abstractions

Josh: Portability to different architectures

compile optimizations
Goal 1

Become familiar with how code “really” runs
Most Important Goal

Have Lots of Fun!
Course Summary
Language to compile: Python 2
Lots of cool features: first-class functions
"functions are values",
objects
Usually, the course structure—Not Us

We will build a compiler in this week!
Do a whole compiler for more and more complex languages

Python

arithmetic expressions
simple assignments
print( )
input( )

HW1 P1
HW2 Parsing P0
HW3 Reg Alloc P0
HW 4 P1
Requirements
Assignments

- Homework projects (in pairs, ~every 2 weeks)
- Exercises (“Exam Prep”)
- Reading and Active Participation
- Midterm Exam and Final Exam
- Final Project (groups of 3-4)
Online Discussion: Piazza

• “Begin Active”
  - Post $\geq 1$ substantive comment, question, or answer each week

• Take a moment to reflect upon the day’s reading or class discussion

• Replaces E-Mail
  - Course announcements
  - Questions for course staff
Final Project

• Often, build on your basic compiler
• Options:
  - Implementation project (undergraduates)
  - Research project (graduates)
• Write a ~5 page paper
• Give a ~15 minute presentation
• Your opportunity to customize the course to your interests!
• Projects in groups of 3-4
Website