CS 380C — Spring 2015	Handout #1
Compilers	**January 21, 2015**

	Professor	Teaching Assistant
Name:	Calvin Lin	Jia Chen
Email:	lin@cs.utexas.edu	jchen@cs.utexas.edu
Office Hours:	Mon/Wed 3:30-4:30	Tue 3:30-4:30, Thu 3:30-4:30
Office:	GDC 5.512	GDC 5.440

Course Objectives

To learn the basics of static analysis and transformation techniques, to understand how static analysis can play a role in many different aspects of systems, and to prepare students to do research in compilers or related fields. Students will learn this material by building new pieces of an existing compiler, reading papers, and proposing and completing a course project.

Prerequisites

Facility with C++. Reasonable programming skills. Good English communication skills. Time and motivation.

Reading Material

- Assorted technical papers. (There is no textbook.)

Lecture Topics

We will cover roughly the following topics, though this list is subject to change.

Part 1: Basics

- Control flow analysis
- Dataflow analysis
- Reuse optimizations
- Static Single Assignment

Part 2: Interprocedural Analysis

- Motivation
- Pointer analysis
- Dimensions of the analysis space
- Flow-Sensitive analysis
- Flow-Insensitive analysis
  - Subset-based
  - Equality-based
- Context-Sensitive analysis

Part 3: Modern Uses

- Modern goals

- Correctness and security
- Dynamic optimization
- Object-oriented languages

Part 4: Classical Uses

- Register allocation
- Instruction scheduling
- Locality and parallelism
- Dependence analysis

Programming Assignments

There will be approximately 4 programming assignments that will build upon the *LLVM* compiler. LLVM is written in C++, makes heavy use of the Standard Template Library, and runs on the CS Department's Linux boxes. The following is a tentative list of programming assignments, although the list is likely to change.

- Assignment 1: Introduction to LLVM
- Assignment 2: Local Optimizations
- Assignment 3: Data-flow Analysis Framework
- Assignment 4: Global Analysis and Optimizations

Course Projects

The course project allows students to explore one area of compilation in some depth. These projects will typically involve implementation, but with proper justification could take on many flavors—including a careful survey, an implementation, or an experiment—and they can be done individually or in groups. However, all course project ideas must be approved by the instructor, and students who do not propose a suitable project will be assigned one. Ideas for course projects will be circulated shortly.

Reading Assignments

There will be 6-9 assigned papers to read. These will sometimes reinforce the lectures and sometimes expose you to new ideas. All reading assignments are fair game on the exam.

Exams

There will be a comprehensive final exam.

Communication Skills

Good communication skills, and in particular good English writing skills, will be important to succeed in this course. For each programming assignment, students will write a report, and the assignments will be graded on clarity and presentation as well as program correctness. In addition, good communication will be needed skills to propose, narrow, and define the course projects.

Grading

Assignments and Homework:	35%
Project:	30%
Final exam:	30%
Class participation:	5%

Assignment 0

Due: January 28.

Part 0: Join our Piazza group so that you can participate in online discussions about the course and the assignments: `piazza.com/utexas/spring2015/cs380c`

Part 1: Read and understand our class rules on Academic Dishonesty (see below) and the University's rules on plagiarism:
`http://deanofstudents.utexas.edu/sjs/acadint_plagiarism.php`

If you have any questions about academic dishonesty or plagiarism, please post to the Piazza page (use Piazza's private message facility if you do not want your questions to be made public).

Part 2: Send a brief email message to both the professor and TA:

(a) Please tell us your academic status (eg, 2nd year PhD student in CS). If you have an advisor and/or a research topic, please let us know what they are.

(b) Let us know if you have read and understood the University's rules on plagiarism. If you still have questions that were not answered by the Piazza discussion, please let us know.

(c) If you have any specific reason for taking this course or any particular topics that you would like to better understand, please let us know.

Part 3(a): As specified in the Piazza post, download the virtual machine image from `http://www.cs.utexas.edu/~jchen/380c.ova`. This virtual environment will be the canonical environment for the entire class. It provides a prebuilt LLVM development environment and the clang compiler so that you do not have to worry about setting up the infrastructure yourself. If you are happy with the performance of this virtual machine, you can skip part 3(b). If instead you want better performance, go to part 3(b).

Part 3(b): Build and install LLVM libraries and the clang compiler on your own Linux machine. We will use LLVM version 3.5.1 (or 3.5.0) in this class. Instructions can be found on `http://llvm.org/docs/GettingStarted.html` if you use GNU Make or on `http://llvm.org/docs/CMake.html` if you prefer CMake. While reading the guides, please note that you do not want to check out any codes from the svn repository as the guide says; instead, source codes should be downloaded from `http://llvm.org/releases/download.html`.

There is nothing to turn in for Part 3; just let us know if you have problems.

Open/Closed Door Policy

Feel free to stop by any time my door is open, which will be most of the time. If my door is closed, *please* do not knock unless you have a scheduled appointment.

Academic Dishonesty

Understand the difference between cheating and collaboration. Allowable collaboration is encouraged. **Cheating will lead to failure of the course.**

There are many **examples of cheating**, but these include accessing another student's account, looking at someone else's code, copying or downloading someone else's code, or allowing others to copy or access your code. Of course, this means that you should not look on the Internet for code to solve your problems.

Examples of allowable collaboration include discussions and debates of *general* concepts (including C++, STL, build systems, etc.) and solution strategies. **Examples of unallowable collaboration** include discussions and debates of implementation details such as code structures and what APIs to use. A good way to ensure that you are collaborating fairly is to follow the Gilligan's Island Rule:

The Gilligan Island Rule
You are free to discuss a problem with others[1], but you may not bring from these discussions any written or electronic notes. After the meeting, engage in a half-hour of mind-numbing activity, such as watching a rerun of Gilligan's Island, before you resume work. This rule ensures that you are able to reconstruct what you learned during your discussion using only your own brain.

Always cite your collaborators with a brief explanation of the degree of collaboration (eg. "Susan and I discussed various approaches to testing our code." eg. "I am using the dominators algorithm described in Chapter 7 of Muchnick's book").

Code Reuse. The code you submit should be your own. You may include LLVM headers and look into their implementation file for ideas, but **do not copy any code from LLVM into your own source**, and do not use any APIs that are private in the sense that they are not accessible through the LLVM include files. We may occasionally ban parts of the LLVM APIs because they will trivialize some of the assignments. Of course, you should **not include any file from any banned parts of the LLVM library**.

Using Outside Learning Material. Materials from the internet should only be used for educational purposes. Thus, you can read about C++ smart pointers (and these examples could well contain code), but you must not copy any code or be looking at any of this code when writing anything that you turn in.

If you have any doubts about what is allowed, please ask the instructor or TA.

[1] Of course, the rule about not looking at anyone else's code still applies.