CSC 431: Compiler Construction

Instructional Information

Professor: Aaron Keen
E-mail: akeen@calpoly.edu
Office: 14-228
Office hours: M: 2-3pm, T: 2-4pm, W: 3-4pm, F: 2-3pm
Course Webpage: `http://www.csc.calpoly.edu/~akeen/courses/csc431`

Lecture Time and Location

Lecture: MWF 9:10am – 10:00am, online

Lab: MWF 8:10am – 9:00am, online

Learning Environment

I believe in a supportive learning environment wherein every person deserves respect, every question deserves an answer, and we all work together toward a goal of improved understanding and life-long learning. All members of this class are expected to contribute to a respectful, welcoming, and inclusive environment for every other member of the class.

There is an old adage that encourages those with questions to ask because others may have the same question; you do not need some imagined quorum in order to validate your question or your request for assistance. If you have a question or need assistance, that alone is sufficient. You belong here and your participation in the course is not only welcomed but encouraged.

Personal Well-being

There are times when coursework and non-academic obligations may conflate to create a seemingly unbearable situation. During these times, your coursework may suffer and you may opt to not complete a milestone by the posted date; I understand this and you need not apologize for it. If this happens, do not get discouraged. The course is structured such that you can choose the milestones on which you wish to focus and you can receive significant partial credit for milestones submitted by the end of the quarter. I am here to assist you; just ask for help.

Course Objectives

- Explain the design and implementation of a compiler.
- Implement features related to the "back-end" of a compiler.
- Explain and implement code transformations.
- Articulate what an optimizing compiler can do and the implications for how you write code.

Prerequisites: CSC 430

Texts

The recommended course textbook is *Engineering a Compiler* by Cooper and Torczon or *Compilers: Principles, Techniques, and Tools* by Aho, Lam, Sethi, and Ullman. Supplemental materials will be linked from the course webpage.

Webpage

Clarifications, changes, etc. regarding the class and assignments will be posted to the course webpage (http://www.csc.calpoly.edu/˜akeen/courses/csc431). Read it regularly, especially near when assignments are due. You are responsible for any announcements posted on the course website.

Activities

Class Participation

The lectures are for your benefit. You should ask questions when you have them. Use lecture time to discuss general approaches to the project.

Lecture Comprehension Quizzes

There will be multiple low-risk quizzes to assess comprehension of lecture material. These will generally be posted on the day of each lecture with an extended period of time to complete them.

Project

There is one large project with multiple milestones. The due dates for the milestones are listed on the schedule. You are allowed, but not required, to work with a single partner on the project.

Each milestone must be demonstrated in lecture or lab on the day that it is due to earn full credit. Milestones are graded based on a somewhat subjective measure of how "complete" the required functionality is. Milestone completion may be demonstrated by the "final demonstration" date for significant partial credit.

You must submit your final project by the date specified on the schedule. This submission must include all of your source code, instructions on how to build your project, and instructions on using your compiler.

Choose Your Own Adventure – Project Paths

Each final project submission will be evaluated based on the ability to generate valid code for the provided acceptance tests. There are multiple legitimate paths to completing the project, with varying learning outcomes and, as such, varying potential final course grades. These are summarized as follows.

Name	Description	Maximum Grade
LLVM-only	Register-based LLVM - without optimizations (1–3)	C
LLVM-opt	Register-based LLVM - with optimizations (1–3, 6)	B
Unoptimized	Assembly from register-based LLVM - without optimizations (1–5)	B
Optimized	Assembly from register-based LLVM - with optimizations (1–6)	A

Paper

Each group will submit a paper detailing the design and implementation of their compiler project. At least half of the grade for the paper will depend on the presentation of some performance analysis of the code generated by the group's compiler. Further details are provided on the course website.

Grading

The percentage breakdown for the course grade is as follows.

Activity	% per	% total
Quizzes		5
Milestones	5	30
Final Submission	50	50
Paper	15	15
	Total	**100**

Collaboration and Cheating

Students may work in pairs on the project. Each student/pair is expected to complete their own project without collaboration with others.

It is fine to talk with others about general approaches to the project, *but* each student/pair is to develop their own solution; collaborative efforts beyond a recognized pair are **not** allowed. Students/pairs are not to view any other student's code or exchange code in any form (hardcopy or electronically).