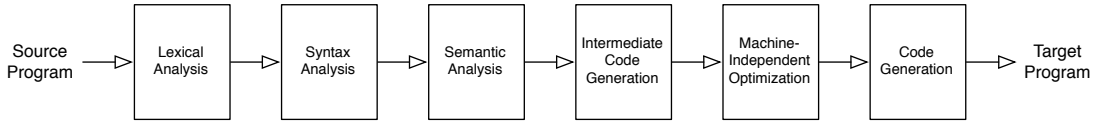
Syllabus

CSCI 434T
Fall, 2015

Compiler Design



Instructor	Prof. Stephen Freund
Office	TPL 302 (top floor of Physics, facing West College)
Phone	597-4260
Email	freund@cs.williams.edu

Web Page `http://www.cs.williams.edu/~freund/cs434/index.html`

Texts

The primary text book for this course is *Compilers: Principles, Techniques, and Tools* by Aho, Lam, Sethi, and Ullman. Additional reading material and resources will be made available to you.

Course Objectives

This tutorial covers the principles and practices for the design and implementation of compilers and interpreters. Topics include all stages of the compilation and execution process: lexical analysis; parsing; symbol tables; type systems; scope; semantic analysis; intermediate representations; run-time environments and interpreters; code generation; program analysis and optimization; and garbage collection. The course covers both the theoretical and practical implications of these topics. As a project course, students will construct a full compiler for a simple object-oriented language.

Tutorial Meetings

This course will be taught as a tutorial. Each of you will be assigned to a group of two students. Your group will normally meet with me each week to discuss the readings and exercises I assigned for the week. These exercises will focus on "problem set" questions, small programming exercises, primary literature, design of your projects, and possibly even code reviews.

In the "canonical" tutorial format, one student in each group is primarily responsible for the presentation each week. Because of the cumulative nature of the material we will be considering, I will expect each of you to come to each meeting fully prepared to present summaries of the readings and solutions to the exercises. To compensate for the higher workload such an approach entails compared to the "canonical" tutorial, you are encouraged to work on the assignments with others taking the course. To make our meetings more interesting, however, I ask that you not work as one large class group. (And I may also ask you not to work with your tutorial partner on some weeks.)

The tutorial format naturally emphasizes learning more than the specific material covered by the course. In particular, this format can have the very positive effect of encouraging you to develop your ability to learn new material independently and to improve your ability to present your thoughts orally. Therefore, in completing the assignments, you should do more than prepare written solutions. You should take some time to think about how you will present your solutions. I do not mean that you should prepare a lecture. Rather, take the time to look over each reading and decide what the major points are. Do the same with the exercises. I will ask you to submit polished, written solutions to some of the problems the day after your meeting.

Programming Project

As a project course, there will also be a substantial semester-long programming project. The overall goal of the project is to build a fully functional compiler for a small but fairly complete subset of Java. The project will be done in groups of three. You may choose your own project groups, although I may choose to reorganize them as the semester progresses.

Programs should be turned in electronically by the due date. Each group may use a maximum of three free late days during the course of the semester. A late day permits you to hand in an assignment up to 24 hours late, without penalty. In order to use a late day, you must send to me *before the deadline* a status update that includes a description of what has been completed, what functionality still needs to be implemented, and an outline of what steps remain.

Late days are designed to give you some flexibility, but be warned that the programming projects are substantial and cumulative — falling behind or failing to complete one project will substantially impact the follow-on assignments. Use late days if you need to, but plan ahead and start early. You will not finish these assignments if you wait until the last minute.

Once those late days are exhausted, late assignments will be penalized severely.

All programs will be graded on design, documentation, style, correctness, and efficiency. We will use the Unix lab computers for the programming assignments.

Grading

Grades will be determined roughly as follows: Homework/Meetings: 50%, Projects: 50%.

Honor Code

Collaboration on problems is permitted and even encouraged, as noted above. However, copying of solutions is not. The work you hand in should be your own. A good rule to follow is to work through the problems with others, taking only rough notes. You should then write your solutions independently, referring to your notes as little as possible. The idea is to understand each solution well enough that you can reconstruct it by yourself. In addition, you should write on each assignment the list of people with whom you collaborated. The same applies to programming — feel free to discuss general ideas with others, but the code your submit should have been written entirely by you or your group. I will occasionally give more detailed collaboration guidelines on some assignments.

Uncredited collaborations will be considered a violation of the honor code and will be handled appropriately. For a full description of the Computer Science Honor Code, please see

`http://www.cs.williams.edu/~freund/honor.html`

If in doubt as to what is appropriate, do not hesitate to ask me.

Tentative Schedule

The course is structured roughly as follows. This will undoubtedly change.

Lexical and Syntax Analysis	3 weeks
Abstract Syntax Trees and Semantic Analysis	2 weeks
Intermediate Representations and Code Generation	3 weeks
Optimization	3 weeks
Advanced Topics	1 week

Be sure to check the web site for an up-to-date schedule.