CSCI 390 – Spring 2013
Compiler Construction

Ben Coleman
coleman@cs.moravian.edu
214 PPHAC

Office Hours: TR 9:00-11:00
or by appointment
Office Phone: 610-625-7781

Course Description

Broadly speaking, a compiler is a program that translates input from one representation to another while verifying its form. Although traditionally this translation is from a programming language to machine language, compiler theory is now used in an amazingly wide variety of applications. RTF documents, email headers, XML files, and web server logs are all examples of human-readable formats that can be processed and verified quickly and easily using compiler technology.

We will study the fundamental components of all compilers: lexical analysis, parsing, semantic actions, intermediate representation trees, and code generation. Throughout the semester, we will develop a compiler for a small, but fully function subset of Java. Because this will be a substantial project, a significant portion of the course will emphasize techniques to design, test, and debug the project.

Course Goals

Upon completion of this course, a successful student will be able to:

- Describe the stages of compilation and how they apply to modern, non-traditional applications.
- Describe the process of generating a lexical analyzer from the regular expressions that represent the language tokens.
- Construct lexical analyzers and both top-down and bottom-up parsers using parser generator tools.
- Implement recursive data structures using the composite pattern.
- Generate intermediate representation trees and use the visitor pattern to perform appropriate actions on the tree (semantic analysis, optimizations, and target-language generation).
- Design, implement, and test a symbol table and other supporting data structures.

Required Text

In addition to the following required text, supplementary readings will be given periodically during the semester.

- *Modern Compiler Implementation in Java – 2nd Edition* by Andrew W. Appel

You should expect to spend about an hour before each class session working through the readings. This means reading the text for detail, studying the syntax for new language features, and working to learn vocabulary – not just skimming through the material before class.

Graded Material

- **Homework** - The goal of homework problems is for you to practice using the current course content and to explore the topics in more detail. Problems will be assigned nearly every class session and generally will be due the next class.

  Homework problems will be graded on a scale between zero and three:

  - **3**: You completed the problem perfectly or nearly perfectly.
  - **2**: Your solution had non-trivial problems.
  - **1**: You tried the problem, but either didn't get very far or made serious mistakes.
  - **0**: You failed to turn anything in for the problem.

  At the end of the semester, your average homework problem score will translate into an actual letter grade as follows:

≥ 2.5	A
≥ 2	B
≥ 1.5	C
≥ 1	D
< 1	F

  Essentially, this scale means that you must earn threes on at least half of the problems to be in the A range (with the remainder of your scores being twos). Plusses and minuses will be used within each range.

- **Programming Assignments** – The book is structured such that each chapter has its own programming project. The projects (except the first) build on each other so that you produce a functional compiler at the end of the course.

  The first project is designed to help you gain familiarity with Java and unit testing. We will use an automated testing tool that runs your code against my suite of tests and measures the coverage of the tests you submit with the code. You are done when all my tests pass and your code has 100% coverage. Your grade will be based on how many submissions are required to reach this state.

  The other six or seven portions of the project have you construct a working compiler. Grades for these assignments will be based on functionality and quality of testing. At the end of each assignment, I will meet with you so that you can demonstrate your progress *on department*

equipment. Grades will be assigned immediately, but can be retroactively increased by later demonstrating that additional functionality was added. In theory, it is possible to earn a perfect score on each assignment, but time constraints will likely make significant post-grade improvements impossible.

- **Tests** – There will be two tests during the semester. One will be given after we complete Chapter 3, and the other after we complete Chapter 6.
- **Final** The final will be cumulative and will be held on Tuesday, April 30 beginning at 8:30 A.M. Any change to the final exam schedule must be approved by both me and the dean of students.

Grade Determination

- (25%) Homework
- (45%) Programming Assignments
- (20%) Tests
- (10%) Final

All grades will be calculated on the standard scale using pluses and minuses.

Course Policies

- **Late Policy** – I understand that life sometimes gets in the way of getting work done. Consequently, late assignments will be accepted without penalty in the class after the assignment was due. However, this policy should not be used as a crutch, and if you frequently use it I will deduct from your grade. After the next class session, late work will not be accepted unless there are exceptional circumstances.
- **Extensions** – In a similar vein, I am generous with extensions on work if you approach me *before* the day the assignment is due.
- **Absences** – Your attendance is expected at each class meeting, but I understand that students occasionally get sick, have obligations outside Moravian, and even over sleep. If you do miss class, please send me an email explaining your absence – preferably before the class session. Regardless of your reason for missing class, you are responsible for the contents of reading assignments, handouts, class activities, and class email.
- **Academic Honesty** – Except on tests, you are *encouraged* to discuss the material and work with other students in the course. Specifically, on homework and programming assignments you may discuss any portion of the assignment with your fellow students. This policy does not allow you to copy another student's work verbatim – you must produce your own code or write-up of the material. Work together to learn the concepts, but keep in mind that you are ultimately responsible for the material on the tests.
- **Disabilities** – If you have a disability that may affect your performance in this course, please contact me immediately to discuss academic accommodations.

Schedule

We will cover the first eleven chapters of the book in order:

- Chapter 1 Introduction
- Chapter 2 Lexical Analysis
- Chapter 3 Parsing
- Chapter 4 Abstract Syntax
- Chapter 5 Semantic Analysis
- Chapter 6 Activation Records
- Chapter 7 Translation to Intermediate Code
- Chapter 8 Basic Blocks and Traces
- Chapter 9 Instruction Selection
- Chapter 10 liveness Analysis
- Chapter 11 Register Allocation

The details of this syllabus are subject to change based on our progress through the material.