MC 911 - Compiler Construction Laboratory

Marcio Machado Pereira (mpereira@ic.unicamp.br)
Institute of Computing, UNICAMP

February 7, 2017

Course Information

Welcome to MC 911 Institute of Computing course in the practical aspects of compiler construction. We have an exciting semester ahead of us loaded with that wonderful combination of tools and coding that makes compilers so cool. Class meets Wednesday in CC02-CC03 from 2:00PM - 6:00PM.

The goal of this course is to write a complete compiler for Lya Programming Language, designed exclusively for this course. Lya has the essential features of a realistic programming language, but is small and simple enough that it can be implemented in a few thousand lines of code. You can implement it in Python, C++ or Java but is highly recommended to do it in Python.

While you don't need to be a Python wizard to complete the course, you should feel comfortable working with the language, and in particular should know how to program with classes. If you'd like a Python refresher, there are a bunch of useful links on the website to help you get up to speed. I hope that the use of Python in this course doesn't deter you from taking it. I will also hold a Python review session in the first lecture.

This course will revolve around two programming projects that taken together form a complete working compiler. You will learn how to put the presented techniques in MC910 course into practice and will study all the details involved in the implementation of a "real" compiler.

The first project requires you to implement a scanner, a parser, and some semantic actions, like type checking, for the Lya Programming Language, specified by Lya grammar (https://iviarcio.wordpress.com/lya-bnf-grammar). Study the specification of Lya grammar carefully. To complete this first project, you will use the PLY (http://www.dabeaz.com/ply), a Python version of the lex/yacc toolset with same functionality but with a friendlier interface. For those who prefer to write the compiler in C++, you should use the lex/yacc toolset (http://dinosaur.compilertools.net)and for those who choose java, you guys can use antlr (http://www.antlr.org).

The second project you will extend your semantic processing and deals with code generation for LVM, the Lya Vitual Machine, specified in LVM document (https://iviarcio.wordpress.com/Lya-virtual-machine). The virtual machine separates computer architecture from language issues and simplifies design and portability. Before you start, carefully study the specification of the LVM in order to become familiar with the run-time data structures, the addressing modes, and the instructions.

Also, the second project requires you to implement a small interpreter for LVM. There are many ways of implementing a virtual machine like LVM. A natural and efficient way is through macro-assembly: each LVM instruction can be defined by a macro-instruction (sequence of machine instructions) of the target computer. This approach would result in a quite realistic code generation process. Another approach, particularly convenient for teaching purposes, is an implementation through a simulator or interpreter. I will try to give you a hand by giving you some of the pieces prewritten and littered your code with debug printing and helpful assertions. By the time you're done, you'll have a pretty thorough understanding of the runtime environment for Lya programs and will even gain a little reading familiarity with virtual machines.

Overall, your grade for this course will be determined as

Project	Weight	Tentative Schedule
Lexical, Syntatic & Semantic Analysis	50%	may/03
Code Generation & Virtual Machine Implementation	50%	jun/28

Readings

The course website is https://iviarcio.wordpress.com and it's loaded with resources for this course. There, you'll find all the handouts for this course, along with lecture slides and any code examples that I write in class. It also has useful links for learning more about the tools we'll be using (Python Lex and Yacc), the Python programming language, and compilers in general.

In addition to the handouts, there are two books that I strongly suggest that you pick up. Neither of these textbooks are required for the course, but they're great resources. Those texts are

Compilers: Principles, Tools, and Techniques, Second Edition by Aho, Lam, Sethi, and Ullman. This book, affectionately called the Purple Dragon Book by its readers, is an excellent introduction to the practical and theoretical techniques necessary to build a fully- working compiler. The Compilers book goes into great detail about the techniques we'll be covering, then continues onward to explore more advanced concepts.

Modern Compiler Implementation in Java, Second Edition by A. W. Appel and J. Palsberg. Cambridge University Press, 2002. The book is widely adopted for undergraduate courses on compiler design, covering all of the phases of compilation in a clear, readable fashion.

Office Hours

I will be holding office hours throughout the semester. I will be sending out an email soon where you can let me know what times would work best for you, and will try to schedule office hours to ensure that there's a time that fits nicely into your schedule.

Late Policy

The programming projects in this course each implement one step in the compilation process, and consequently the second assignment builds off of the first one. This means that if you start falling behind on the first one, it can be extremely difficult to get caught up again. That said, I completely understand that you may need some extra time to complete the first assignment. In that vein, you are allowed five calendar late days that you can use on first assignment without penalty. Although you don't need to let me know that you're using them, I would appreciate a heads-up so that I know to expect your solution late. However, once you've used all five late days, your assignment will be penalized 10% per hour late, so please do try to get it on time! For logistical reasons, there are no late days allowed in the second scheduling assignment.

Honor Code

This one should be pretty simple. Don't turn in someone else's project as your own, or share your solution with other students. You should feel free to talk about the programming projects in a group, but all the work you submit should be your own. That is, you should be implementing your programming projects on your own. If you do discuss ideas with other students, that's fine, please make a note of it in your submission. An attempt at fraud in any project will result in a final note f = 0 (zero) for all involved, notwithstanding other sanctions. This course has allowed students to work in pairs on the programming assignments. However, at least for the time being, this semesteer you must complete all of the assignments individually. I may review this policy later in the semester, in which case I'll make an announcement to the class.

I think this is definitely the most fun of the projects; its an awesome feeling when you finally get to the stage where you can compile Lya programs and execute them. I hope you enjoy the course!