Software Atelier 5
Languages and Compilers
Fall 2013

Administrivia:

Instructor:	Nate Nystrom
Office:	SI-203
Email:	`nate.nystrom@usi.ch` (use rarely; use staff email instead)
Assistant:	Ilya Yanok
Staff email:	`usi-compilers@googlegroups.com`
Class Time:	Tuesday, Thursday, Friday, 13:30–15:15
Class Location:	SI-006 (Tu), SI-013 (Th, Fr)
Web:	`http://inf.usi.ch/nystrom/teaching/compilers/fa13`
Moodle:	`http://www2.icorsi.ch/course/view.php?id=2722`

Objective: In this course you will learn how programs written in high-level languages are executed on modern hardware. Understanding how languages are implemented is useful for reasoning about program behavior and performance. A secondary goal of the course is to expose students to the principles, techniques, and tools used to construct compilers and interpreters.

Content: The course will cover both the theory and practice of programming language implementation. Topics include compiler structure, lexical and syntactic analysis (parsing), types, semantic analysis, program representations, data-flow analysis, register allocation, optimization, and compiler construction tools. Students will implement a compiler for a high-level programming language on modern hardware.

Teaching mode: Language implementation is a large topic. It's often complicated. There is a fair amount of theory in this course. This means there will be more lectures than is usual for an atelier.

The course is heavily project-based. Students will implement a compiler for a high-level programming language on modern hardware. Students will also do written assignments to understand programming language and compiler principles Students are expected to read the assigned readings before class. During class sessions we will discuss remaining questions and problems.

In general, I will lecture for a few days about what you need for the next phase of the project, then give you some time in class to work.

References: I will post lecture notes online. You are expected to read the lecture notes.

Much of the course will follow a draft of a book by Jeremy Siek and Evan Chang. A PDF of the book is available on the course web page. The project is based on the project described in the book, although there are some major changes.

There are two recommended textbooks for reference.

- Compilers: Principles, Techniques, and Tools, 2nd edition by Aho, Lam, Sethi, and Ullman, 2006
- Modern Compiler Implementation in Java, 2nd edition, Andrew W. Appel and Jens Palsberg, 2002

Evaluation: The course centers on developing a compiler for a subset of Python. This will be implemented in groups throughout the entire semester. In addition, students will do exercises in class and are expected to participate in classroom discussions and to make presentations to the class on aspects of web programming. Grades will be weighted as follows:

Assignments and exercises	10%
Compiler project	60%
Exam 1	15%
Exam 2	15%

Late policy: Each phase of the project builds on previous phases. It is important that you implement each phase correctly. After you submit code, you must walk through the code with me and show me that it works. If it doesn't work I will ask you to fix the bugs and come back the next day. If it works on the first day, you get full credit. Each additional day, you lose 10

If you are going to miss class for some reason, tell me beforehand. If you tell me early, I can accomodate you.

Cheating and plagiarism:

> Honesty may not be the best policy, but it is worth trying once in a while.
>
> — *Richard Nixon, 1970*

Do not represent others' work as your own. This is plagiarism. Do not copy others' work without citation.

You are free to discuss the project and assignments with other students. But, you must write up your assignments on your own. Any code you write must be your own, unless you obtain prior permission from the course staff to use other code (e.g., code found on the web). Students found to be cheating will receive -100% penalty on the assignment or exam.