CS 577 Compiler Construction (Modern Language Processors) Spring 2002

Instructor:
Andrew Tolmach
120-23 FAB
(503) 725-5492
email: `apt@cs.pdx.edu`
Office Hours: MW 4-5pm & by appt.

Course web page: `http://www.cs.pdx.edu/~apt/cs577`

Description

An introductory graduate-level course on modern techniques for programming language compilation and interpretation. We will focus on the implementation of the Java language. Topics will include: the object-oriented runtime model, virtual machines, native code generation, garbage collection, optimization, and feedback-directed compilation.

Prerequisites

An undergraduate compiler course, such as CS321/322, or equivalent; familiarity with an object-oriented programming language such as Java or C++; strong programming skills.

Readings

Readings will be taken from research and survey papers made available on the web. There will be one or two papers assigned per week.

There is no required textbook, but a number of useful optional texts will be recommended and made available for borrowing.

Requirements

There will be a number of homework assignments, a project, and a take-home final exam. The homework assignments will be short exercises intended to make sure that all students get some hands-on experience with the innards of Java compilers and tools. The final exam will cover the required readings.

The course grade will be distributed as follows:

Project	50%
Homework	20%
Final Exam	30%

Although it will not be formally assessed, class participation is strongly encouraged, and may affect borderline grades.

Project

Each student will select a project of his or her choice, in consultation with the instructor. Possible projects include: implementation work on a real Java compiler, prototype implementations on a ”toy” compiler, performance analysis of existing implementations, or written surveys of the research literature on particular topic. Here are some more specific examples (just as illustrations):

- Compare the quality of generated code produced by three different JVM compilers (e.g., Sun’s HotSpot, IBM’s commercial compiler, IBM’s Jikes compiler) on a large Java benchmark suite.

- Implement a new garbage collection algorithm for the Jikes RVM, and compare its performance with that of the existing collectors on that platform.
- Add a peephole optimizer to the Kaffe JIT, and see how it affects overall performance.
- Write a paper surveying the research literature on fast register allocation algorithms suitable for use in JITs.
- Write a paper comparing the JVM and Microsoft's .NET Common Language Runtime platform.

All projects, even those whose primary product is code, must include a written summary of results. In addition, students are strongly encouraged, though not required, to present their project results to the rest of the class at the end of term.

The scope and difficulty of acceptable projects can vary widely, to accomodate students with varying levels of interest and available time. The difficulty of the project will be factored into the grade: i.e., you'll need to do an excellent job on a small project in order to get the same grade as for doing a merely decent job on a challenging project.

For large projects involving substantial implementation work, it may be appropriate for students to work in teams of two; special approval of the instructor will be required.

A larger list of project suggestions and further guidelines on projects will be issued later.

Computing Facilities

Some of the homework exercises may require access to software that will be installed on CS department machines. Otherwise, you are free to work on whatever machines are convenient for you and your project.

Individual Work

All homework assignments, projects, and exams must represent your own, individual work (except for approved team projects). It is permissible to discuss assignments with other students, but the solutions must be recognizably your own. *Do not, under any circumstances, copy another person's program or text and submit it as your own.* Writing code for use by another or using another's code or text in any form (even with their permission) will be considered cheating, the penalties for which are described in detail in the CS Department's *Graduate Handbook*. In particular, cheating on an assignment or exam will result in an automatic zero grade for that piece of work.

Tentative Schedule

dates	*topics*
Apr 1 & 3	Compiler Architecture; Modern Languages
Apr 8 & 10	Java Virtual Machine and Bytecode
Apr 15 & 17	JVM Interpreter Implementation
Apr 22 & 24	Garbage Collection
Apr 29 & May 1	Generating Native Code
May 6 & 8	Memory Optimizations
May 13 & 15	SSA-based Optimizations
May 20 & 22	Dynamic loading and Feedback-directed Optimization
May 29	Portable Code Verification
Jun 3 & 5	Project Reports
Jun 12	(Wed) 12:30-14:20 Final exam slot (avaiable for project reports)