CS 577 Modern Language Processors Winter 2004

Instructor: Andrew Tolmach
120-23 FAB (503) 725-5492
email: apt@cs.pdx.edu
Office Hours: MW 11am-noon & 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 Java and similar languages. Topics will include: program optimization, runtime system implementation, garbage collection, and virtual machine architectures.

Prerequisites

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

Readings

The textbook will be Andrew Appel, *Modern Compiler Implementation in Java, 2nd ed.*, Cambridge University Press, 2002. (It is also fine to use earlier editions, including the ML-based edition, but be careful to check out the errata on the author's web page.) We will cover most of the "advanced" topics in the book.

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

Requirements

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

- Implement some standard flow-based optimizations in Appel's functional IR framework, and compare the results to the usual SSA implementations.
- Add a peephole optimizer to the Kaffe JIT, and see how it affects overall performance.
- Write a paper surveying the research literature on compilation to minimize power consumption of generated code.
- 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 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 larger projects involving substantial implementation work, you are *encouraged* to work in small teams; approval of the instructor on team makeup will be required.

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. Cheating on an assignment or exam will result in an automatic zero grade for that piece of work, and the initiation of disciplinary action at the University level.

Disabilities

If you are a student with a disability in need of academic accommodations, you should register with Disability Services for Students and notify the instructor immediately to arrange for support services.

Tentative Schedule

dates	*topics*
Jan 5 & 7	Compiler Architecture; Modern Languages
Jan 12 & 14	Java Virtual Machine and Bytecode
Jan 19	NO CLASS (Martin Luther King, Jr. Day)
Jan 21	JVM Interpreter Implementation
Jan 26 & 28	Generating Native Code; just-in-time vs. ahead-of-time
Feb 2 & 4	Dataflow Analysis
Feb 9 & 11	SSA-based Optimizations
Feb 16 & 18	Memory Optimizations
Feb 23 & 25	Garbage Collection
Mar 1 & 3	Portable Code Verification
Mar 8 & 10	Wrap-up; Project Reports
Mar 17	(Wed) 12:30-14:20 Final exam slot (available for project reports)