CS 577 Modern Language Processors Spring 2005

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, virtual machine architectures and implementation, and garbage collectoin.

Prerequisites

An undergraduate compiler course, such as CS321/322, or equivalent; familiarity with an object-oriented programming language such as Java, C++, or C#; good low-level (C or assembler) programming skills.

Readings

We will cover the advanced sections of *Engineering a Compiler* by Keith Cooper and Linda Torczon (Morgan Kaufmann 2004), and read a number of papers obtainable on the web. For some topics, there may also be class notes by the instructor. There will be one or two chapters or papers assigned per week.

Requirements

There will be a number of homework assignments, a project or paper, 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 or Paper	50%
Homework	20%
Final Exam	30%

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

Project or Paper

Each student will select a project or paper topic of his or her choice, in consultation with the instructor. Students will be encouraged to do implementation projects based on the "toy" Java VM framework provided by the instructor; however, it is also ok to do an implementation project on a different infrastructure (e.g. the McGill Soot compiler, the IBM Jikes RVM, or the Kaffe VM). Papers will be written surveys of the research literature on particular topic. Here are some more specific examples from previous course offerings (just as illustrations; some of these were too ambitious):

- Project: Implementing cache conscious garbage collector in the Jikes RVM.
- Project: Implementing peephold optimizer in the Kaffe JIT compiler.
- Project: Implementing a functional intermediate form for the Soot Java compiler.

- Paper: Stack Inspection in the Java Security Model.
- Paper: Survey of recent concurrent mark&sweep garbage collection algorithms.

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.

(Very) Tentative Schedule

dates	*topics*
Mar 28 & 30	Compiler Architecture; Modern Processors; Modern Languages
Apr 4 & 6	Java Virtual Machine and Bytecode
Apr 11 & 13	Efficient Interpreter Implementation
Apr 18 & 20	Generating Native Code; just-in-time vs. ahead-of-time
Apr 25 & 27	Dataflow Analysis
May 2 & 4	SSA-based Optimizations
May 9 & 11	Memory Optimizations
May 16 & 18	Garbage Collection
May 23 & 25	Portable Code Verification
May 30	**MEMORIAL DAY – NO CLASS**
June 1	Wrap-up; Project Reports
June 6	3:30–5:20 Final exam slot (available for project reports)