CS 322 Languages and Compiler Design II - Spring 2012

Instructor:
Andrew Tolmach
120-23 FAB
503-725-5492
email: apt@cs.pdx.edu
Office Hours: W 1-2 & by appt.
Course home page: http://www.cs.pdx.edu/~apt/cs322

Description

CS321/322 studies, in parallel, the design and implementation of programming languages and the design and implementation of compilers. The course is centered around a substantial programming project: implementing a complete compiler for a realistic language. CS322 concentrates on code generation and runtime organization; we will assume the existence of a compiler front-end, and build a back-end for it. Completed compiler projects will produce machine code that can be run directly on the target hardware. A detailed list of lecture topics may be found below.

Prerequisites

CS321 with a grade of C or better, preferably taken Winter 2012 with this instructor. CS321 and 322 form a connected pair of courses; having completed CS321 in the distant past with another instructor will not necessarily have prepared you for this version of CS322. In particular, this offering of CS322 will use Java (version 1.6+) as the project implementation language.

Substantial experience with the C, C++, or Java programming language and a Unix-like operating system is essential. Some experience with machine-language programming is extremely desirable.

Texts

The required textbook is

• Keith D. Cooper and Linda Torczon, *Engineering a Compiler*, Second edition (2011) is preferred; first edition (2004) is acceptable.

Additional readings from other sources will be handed out in class from time to time.

• Various web materials describing the x86-64 architecture are listed on the course web page.

You'll also definitely want access to a Java textbook. There are many good possibilities; here are two I suggest:

• Ken Arnold, James Gosling, and David Holmes, *The Java Programming Language*, 4th ed., Addison–Wesley, 2005. (Earlier editions don't cover all the features of Java 1.5/6.)

• Bruce Eckel, *Thinking in Java, 4th ed.*, Prentice-Hall, 2006. (The third edition of this book is available free on the web at http://www.mindviewinc.com/Books/downloads.html, but again, it only describes Java 1.4.)

You will also need documentation for the particular version of Java that you choose to use, typically available from http://download.oracle.com/javase.

Lecture notes will be available electronically in pdf format via the course home page, either before or shortly after the lecture occurs. However, lectures may not follow the notes precisely; it is up to you to take additional notes during class.

Project

Over the course of CS322 you will complete the **fab** compiler you began in CS321. You will also build an interpreter for the language. A working version of the front end from CS321 will be available for you to use in place of your own, if you prefer (or if you took CS321 in some earlier term).

You are strongly encouraged – but not required – to work in teams of two on the project assignments. Team members submit a single version of each project assignment, and receive the same grade.

Your compiler must be written in Java version 1.6 or 1.7. Substantial supporting code will be handed out by the instructor.

Programs should be submitted by email; the address and other details on where and how to submit programs will be provided with each assignment.

Exams

There will be one mid-term and a final exam. Both are *closed-book*. Exams will cover topics from lectures and readings, emphasizing material that is not directly relevant to the programming project. Not all the material is covered in the readings, so lecture attendance is important. Exams are scheduled in advance; unless prior arrangements are made, a grade of zero will be recorded for missed exams.

Homework problems may be assigned from time to time as an aid to help you study for the exams; they will not be collected or graded.

Grading

Approximately 1/2 on programming assignments and 1/4 on each of the two exams (including the final).

Programming assignment grades will be determined primarily by observing program behavior on test inputs. Computer programming being what it is, this policy means that "small" errors in your code can have a large effect on your grade. Sample test inputs and a correctly-behaving executable will be provided for you to compare your program against. It is your responsibility to apply these tests *and others of your own devising* before submitting your assignment. We will apply some non-public tests to your programs as well.

Computing Facilities

The project will be implemented in Java and x86-64 assembly language. For several of the assignments, you will need access to an x86-64 processor running a GNU-based unix-like operating system, such as those found on the CS department's linux lab network. For the other assignments, you may develop your project solutions on any machine and OS that supports Java. However, the final version of *all* assignments must work using the CS department's linux lab machines, using Java SDK version 1.6 and (where relevant) the GNU tool chain (assembler and loader).

You should have been given an account on the CS linux lab machines automatically by virtue of registering for this course. If you don't already have an account, go to FAB 82-01 to obtain one.

Files associated with assignments will be made available for download via the course web page.

Mailing List

Important information will be distributed throughout the term via a mailing list called `cs322list@cs.pdx.edu`. To subscribe to the list, visit the course home page and follow the directions from there. Please mail questions to the instructor directly (at `apt`) rather than to this list; the instructor will copy mail of general interest to the list. The list is archived so you can consult old messages.

Schedule

All dates, including assignment deadlines, are tentative (especially the lecture topics).

Date		Asgn	Reading	Topics
Apr	3			introduction; synopsis of backend tasks
	5	1		interpreters; review of attribute grammars
	10		CT 6.2,6.3.1	environments; procedure activations and scope
	12			functional programming
	17	2	skim x86 docs	x86-64 architecture
	19			semantics: operational and axiomatic
	24	1*		semantics: denotational
	26		CT 5.1–4	intermediate representations
May	1		CT 7.1–8	IR generation: expressions and statements
	3	2* 3		IR generation: booleans, backpatching
	8			**Midterm Exam** (in class)
	10		CT 6.3.2,6.4.3,6.5,7.9	compiled runtime organization
	15		CT 6.4.1–2	parameter passing mechanisms
	17		CT 6.3.3–4	implementing object-oriented features
	22	4	CT 6.6	garbage collection
	24	3*	CT 11.1–3	naive machine code generation
	29		CT 8.1–4	optimization principles
	31		CT 9.1–2,13.1–13.4.4	register allocation and flow analysis
Jun	5		CT 12.1–3	instruction scheduling; VLIW architectures
	7	4*		review and wrap-up
	11			**Final Exam** (**Monday** starting at **10:15 am**)

Readings column key: CT = Cooper and Torczon textbook, **second edition**. 2nd ed. 5.1–3 = 1st ed. 5.1–4; 2nd ed. 6.3.4 = 1st ed. 7.10; 2nd ed. 6.4.3 = 1st ed. 6.5; 2nd ed. 6.5 = 1st ed. 6.6; 2nd ed. 6.6 = 1st ed. 6.7; 2nd ed. 11.1–3 = 1st ed. 11.1–2; 2nd ed. 13.1–13.4.4 = 1st ed. 13.1–13.5.4.

Additional readings may be assigned from time to time. Assigned selections should be read *before* the associated date.

Assignments

Programming assignments, listed below, are distributed on the dates the assignment number appears in the schedule above and are due when the starred number appears.

1 interpreter
2 x86-64 machine code exercise
3 preliminary code generation
4 final code generation

Rules

Programs are due *by 1:30 p.m.* on the specified due date, i.e., just *before* class. Late programs are *not* accepted except in extraordinary circumstances, and then preferably by prior arrangement. The project deadlines are there to help you, by forcing you to keep up during the term.

All programming assignments must represent the work of your explicitly identified two-person team. It is permissible to discuss the assignment with other students, but you must develop the solutions yourselves. *Do not, under any circumstances, copy another person's program and submit it as your own.* Writing code for use by another or using another's code in any form (even with their permission) will be considered cheating. In particular, cheating will result in an automatic zero grade for that pice of work, and the initiation of disciplinary action at the departmental and University level.

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.