CS 302 Languages and Compiler Design II - Spring 1999

Instructor:
Andrew Tolmach
PCAT 128A
725-5492
email: `apt@cs.pdx.edu`
Office Hours: MW1-2 & by appt.

Course home page: `http://www.cs.pdx.edu/~apt/cs302`

Description

CS301/302 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. CS302 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 on the accompanying schedule sheet.

Prerequisites

CS301 with a grade of C or better, preferably taken Winter 99 with this instructor. CS301 and 302 form a connected pair of courses; having completed CS301 in the distant past with another instructor will not necessarily have prepared you for this version of CS302.

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

Texts

The required textbook is Aho, Sethi, and Ullman, "Compilers: Principles, Techniques, and Tools," Addison-Wesley, 1986. Additional readings from other sources will be handed out in class from time to time. A good auxiliary reference text is R. P. Paul, "SPARC Architecture, Assembly Language Programming, and C," Prentice Hall, 1994.

Lecture notes will be available electronically in postscript format (and usually also in html) via the course home page.

Project

Over the course of CS302 you will complete the PCAT compiler you began in CS301. You will also build an interpreter for the language. A working version of the front end will be available for you to use in place of your own, if you prefer.

You are *strongly* encouraged to work in teams of two on the projects. Team members submit a single version of each project assignment, and receive the same grade. The first programming assignment of the term is *not* part of the project and must be completed *individually*.

Your compiler must be written in ANSI C or C++. Supporting code handed out by the instructor will be in C; if you use C++, you will need to work at least partly in the C subset of the language. C is described in B. W. Kernighan and D. M. Ritchie, *The C Programming Language*, 2nd edition, Prentice–Hall, 1988, and in numerous other reference books. C++ described in M.A. Ellis and B. Stroustrup, *The Annotated C++ Reference Manual*, Addison–Wesley, 1990, and in numerous other reference books. Also, you will continue to need the PCAT language definition handed out in CS301.

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 2 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

For most of the assignments, you will need access to a Sparc processor, such as those found on the CS department's network. When feasible, you may develop your project solutions on any machine you like, but the final version of *all* assignments must work using the GNU `gcc` or `g++` compiler on the CS department's Sparc machines. If you do not have an account on the department's Sparc network, you should obtain one immediately by following the instructions on The Form, available in class or from the CS department office.

Directory `/u/cs302acc/bin` will contain some useful programs; you should place this directory in your PATH.

Submitting Programs

Programs should be submitted by `mail`ing a "bundle" containing the relevant files to `cs302acc@cs.pdx.edu`. Bundles can be created by the Unix `shar` command, which combines the files given by its arguments into a single shell script that, when run, recreates the files. For example, you could submit the files `makefile`, `lex.c`, and `main.c` as follows:

```
shar makefile lex.c main.c | mail cs302acc
```

Do not send mail other than program submissions to `cs302acc`, as it is not regularly read.

Mailing List

Important information will be distributed throughout the term via a Majordomo list called `cs302list`. You can subscribe to this list by sending a mail message to `majordomo@cs.pdx.edu` with no subject line and a message body consisting of the single line

```
subscribe cs302list
```

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.

Schedule

All dates, including assignment deadlines, are tentative.

Date		Asgn	ASU Reading	Topics
Mar	29	1		introduction; synopsis of backend tasks
	31		Sparc	SPARC architecture
Apr	5			syntax and semantics revisited
	7	2		interpreters; review of attribute grammars
	12	1*	7.1,7.6	environments; procedure activations and scope
	14			expressions and statements
	19		8.1-3	intermediate representations; expressions and simple statements
	21		8.4-6	booleans; backpatching
	26		8.7	statements
	28	2* 3	7.2-4	compiled runtime organization
May	3			**Midterm Exam** (in class)
	5		7.5	parameter passing mechanisms
	10			procedures as parameters; functional programming
	12		9.1-3,9.11-12	code generation; instruction selection
	17	4	9.4-6	naive register allocation
	19	3*	9.7	better register allocation
	24		9.9,10.1-3	optimization principles; instruction scheduling
	26			implementing object-oriented features
	31			**No class - Memorial Day holiday**
June	2	4*		garbage collection
	7		**3:30pm**	**Final Exam**

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 SPARC machine code exercise
2 interpreter
3 preliminary code generation
4 final code generation

Rules

Programs are due *by 3:30 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.

Unless specifically indicated otherwise, all programming assignments must represent your own, individual work. It is permissible to discuss the assignment with other students, but not to share solutions (although you can consult the tutors for help in debugging). *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, the penalties for which are described in detail in the CS Department's booklet *Undergraduate Programs 1998-1999.* In particular, cheating will result in an automatic F for the assignment in question, and a letter to your departmental file.