Version of 16 January 2011

Course Information

Lectures

Tuesdays and Thursdays, 9:30–10:45, in Mitchell Hall 104

Instructor

Darko Stefanovic, office hours Wednesdays 10-11

Teaching assistant

none

Course topics

The course covers introductory topics in compiler construction, including computer organization and architecture, operating system support, code and data layout, memory management, generation of executable code, intermediate representations, simple code optimizations, as well as the traditional topics of syntax analysis.

Despite the catalogue description, students will not have to write a whole compiler. It takes too much effort to write a complete compiler for a complete programming language, and when this is attempted within a semester the relevant lessons tend to get lost under the weight of the programming. Instead, students will implement several components of a compiler, ranging from parsing to code generation, with an emphasis on tools for automated translation. The course will cover the implementation of imperative languages; techniques for functional and logic languages will be left out, as indeed the special techniques for object-oriented languages. The course will cover only very simple program analysis and code optimization techniques. (What we do not cover in this course, we will cover in future editions of CS555 and in special topics classes.)

Prerequisites

There are two essential prerequisites. First, students should be familiar with several high-level programming languages, including representatives of the procedural, object-oriented, and functional programming models, so that they can appreciate the purpose and the tasks of a compiler. Second, students should be experienced programmers able to develop large programming projects implemented in some programming language. This could be formalized as: facility with writing moderate-sized code (CS251/351), basic data structures and algorithms and their implementation (CS251/CS361/CS362), basic computer organization and assembly-level programming (CS341).

Course format

The course will consist of lectures, written homework assignments, and projects. Students enrolled in CS554 (i.e., graduate students) will be given additional work as part of their homework assignments and projects.

Assignments

Midterm exam, final exam (covering the entire course), up to 3 short written homework assignments to consolidate lecture material, 3 programming projects.

Written homework assignments

Homework assignments will be carried out individually. Detailed submission instructions will be given with each assignment.

Projects

Each project will be an implementation of an algorithm or phase in a compiler, or an algorithm or tool used to automatically generate a phase in a compiler. Detailed input/output specifications will be provided. Students will be free to choose any implementation language(s), subject to the constraint that a standalone executable file (runnable on CS machines) must be generated and submitted as part of the solution. (Similarly, the instructor will be free to discuss implementation strategies using any implementation language(s) by way of example.)

Projects will be carried out in teams of two (except for graduate students' additional tasks, which will be carried out individually). Detailed submission instructions will be given with each project.

Textbooks

Required reading

Torben *Æ*. Mogensen, *Basics of Compiler Design*, available at
`http://www.diku.dk/hjemmesider/ansatte/torbenm/Basics/index.html`

General reading on compilers

Keith D. Cooper and Linda Torczon, *Engineering a Compiler*, Morgan Kaufmann, 2003, ISBN-10: 155860698X.

Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman, *Compilers: Principles, Techniques, and Tools*, Addison Wesley, 2nd Edition, 2006.

Andrew W. Appel: *Modern compiler implementation in ML* Cambridge University Press, 1998, ISBN 0-521-58274-1.

Dick Grune, Henri E. Bal, Ceriel J.H. Jacobs and Koen G. Langendoen: *Modern Compiler Design*, John Wiley, 2000, ISBN 0-471-97697-0.

Michael L. Scott, *Programming Language Pragmatics*, Morgan Kaufmann, 2000, ISBN 1-55860-442-1.

David A. Watt and Deryck F. Brown, *Programming Language Processors in Java*, Prentice Hall, 2000, ISBN 0-13-025786-9.

Reinhard Wilhelm and Dieter Maurer, *Compiler Design*, Addison-Wesley, 1995, ISBN 0-201-42290-5.

Niklaus Wirth, *Compiler Construction*, Addison Wesley, 1996 (revised version from 2005 available on-line).

Special topics

Randy Allen and Ken Kennedy, *Optimizing Compilers for Modern Architectures*, Morgan Kaufmann, 2001

Dick Grune and Ceriel J. H. Jacobs, *Parsing Techniques*, Springer, 2nd Edition, 2008 (but 1st edition is available on-line)

John L. Hennessy and David A. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, 4th Edition, 2006, ISBN-10: 0123704901.

Steven John Metsker, *Building Parsers with Java*, Addison Wesley, 2001

Michael Wolfe, *High Performance Compilers for Parallel Computing*, Addison-Wesley, 1995

Grading

You are expected to attend class regularly, read the assigned reading before class, and participate in class discussion. The grade will be determined as follows:

Homeworks 10%

Programming projects 60%

Exams 30% (10% midterm exam, 20% final)

Homework and programming assignment hand-in policy

Written homework assignments are due on the date assigned, no extensions will be granted, and no credit will be given for late homework. Late programming project submissions will be penalized $2n^2$%, where n is the number of days late.

Lecture Plan

- Week 1: Course organization; translation and interpretation.
- Week 1-2: Lexical analysis. RE, DFA, NFA construction; pragmatic issues; tools.

- Week 3-4: Parsing. Top-down; bottom-up; pragmatics; tools.
- Week 5: Names, scope, and binding.
- Week 5-6: Semantic elaboration; attribute grammars.
- Week 7: Representing data types.
- Week 8: Representing control flow.
- Week 9: Representing the procedure abstraction.
- Week 10-11: Intermediate representations.
- Week 12: Code generation.
- Week 13: Instruction scheduling.
- Week 14: Register allocation.

Mailing list

A mailing list will be used for class discussion. It may also be used for administrative announcements.

UNM statement of compliance with ADA

Qualified students with disabilities needing appropriate academic adjustments should contact the instructor as soon as possible to ensure their needs are met in a timely manner. Handouts are available in alternative accessible formats upon request.